Topics in Property Testing over Massively Parameterized Models

Orly Yahalom

Topics in Property Testing over Massively Parameterized Models

Research Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Orly Yahalom

Submitted to the Senate of the
Technion - Israel Institute of Technology

Av 5768 Haifa August 2008

The research thesis was written under the supervision of
Assoc. Prof. Eldar Fischer
in the Department of Computer Science.

Acknowledgements

I am deeply grateful to my supervisor, Eldar Fischer, for his dedicated guidance, encouragement and patience throughout the years. Eldar has always been there for me, and I was tremendously enlightened by our discussions as well as enjoyed them.

I also thank our collaborators: Oded Lachish, Ilan Newman and Arie Matsliah, for the pleasant and fruitful work together.

I am indebted to many people in the Computer Science Department, who have helped me through the course of my research, as well as to my friends inside and outside the Technion, who have made my life easier and happier.

Finally, I thank my dear family, and mostly my mother, Neta, for her endless encouragement and support, both during this research and in all the preceding years.

The generous financial help of the Technion is gratefully acknowledged.

Contents

Abstract

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 property $\mathcal{P}$ and the case where S is "far" from satisfying $\mathcal{P}$. A tester is a randomized algorithm which answers such a question by reading only a small part of the input S, using retrieval procedures that we call queries. Property testing normally deals with very large input structures and/or with costly queries, and thus, the query complexity is assumed to be the most limited resource, rather than the computation time. As decision problems cannot be solved without reading the entire input, our algorithms give approximate results, in the sense that inputs close to satisfying the property $\mathcal{P}$ may also be accepted.

A great deal of research in property testing is dedicated to graph properties. In the standard models, a graph is considered far from satisfying a property $\mathcal{P}$ if many edges must be added or removed from it in order to make it satisfy $\mathcal{P}$. We consider two models of property testing in which a fixed graph G is given in advance as a parameter to the tester. This means that the answer has to be exact with respect to G, and may be approximated only with respect to some additional input, such as a coloring of G. In the first model, we test vertex colorings of G, and in the second model, we test edge orientations of G, following the work of [30]. The graph G itself may not be altered. This definition of the distance function gives us a rich setting for studying various graph properties, devising tests that depend strongly on the structure of the parameter graph but are independent of technicalities such as representation issues.

In the vertex colorings model, we consider several problems of testing for forbidden partially ordered sets. In particular, we study several variations of the convexity property, where a coloring satisfies the property if it induces connected color components. This property is related to the study of phylogenetic trees in genetics. As for the orientation model, we investigate the well studied property of being Eulerian,

and consider general graphs as well as important special cases such as dense graphs and expander graphs. For these properties we give several upper and lower bounds. In particular we show that convexity can be tested with a number of queries independent of the input size. On the other hand, while being Eulerian can always be tested in a sub-linear number of queries, a constant query algorithm does not exist.

Chapter 1

Introduction

In this chapter we provide general background on massively parameterized property testing. We also give a brief overview of this thesis and refer to our papers on which it is based.

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 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. For example, if our inputs are binary vectors of a fixed length n, then a query to an input vector would be an index i, where the answer is the i^{th} element of the vector.

Property testing was initiated in the work of Blum, Luby and Rubinfeld [10], and given a general formulation by Rubinfeld and Sudan [47]. 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 [25]. 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 [46] and [16]. As property testing focuses on algorithms that read only a small fraction of the input, it appeals to many contemporary research areas involving large datasets, such as data mining and bioinformatics. Property testing also emerges in the context of program checking [24, 47, 10], probabilistically checkable proofs [5, 8] and approximation algorithms [25].

Denoting our set of inputs by $\mathcal{I}$, a property $\mathcal{P}$ is a subset of $\mathcal{I}$. We have a *distance function* $\text{dist} : \mathcal{I} \times \mathcal{I} \to [0, 1]$, 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 $\varepsilon \in [0, 1]$ if $\text{dist}(S, S') \leq \varepsilon$. Otherwise, if $\text{dist}(S, S') > \varepsilon$, 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 $\varepsilon, q > 0$, 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 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 $q = q(\varepsilon)$ (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 $\varepsilon > 0$ 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 Standard models for testing graph properties

A model for testing properties in a certain set $\mathcal{I}$ of inputs is characterized by the *distance function* dist on $\mathcal{I}$ and by the type of *queries* allowed to access the input. We are interested mainly in testing of graph properties, i.e., properties defined on a set $\mathcal{I}$ which is a class of graphs.

Goldreich, Goldwasser and Ron [25] introduced the *dense graph model*, in which the input is a simple undirected graph represented by its adjacency matrix. A query in this model is a pair of vertex indices, and the answer is the corresponding entry in the adjacency matrix, which indicates whether these two vertices are neighbors or not. The distance between two graphs in this model is the fraction of adjacency matrix entries on which they differ. The dense graph model has been studied extensively (e.g. [1, 27, 15, 2]). It has also inspired several generalized models, such as for directed graphs [4] and hypergraphs [20].

Although quite natural when the input graph is dense, the dense graph model is in a sense too lenient for some graphs. Considering an n-vertex graph, the distance function of the dense graph model allows adding and removing $o(n^2)$ edges, regardless of the number of actual edges in the graph. Thus, many interesting properties, such as connectivity, are trivially testable in this model, as all graphs are close to satisfying the property.

Researchers have studied several alternative models for graph testing. Notably, in the *bounded-degree graph model* of [26], the graph is assumed to be represented by adjacency lists of bounded length. A query in the model is of the form (v, i), where v is a vertex and i is an index no larger than the maximum degree in the graph. An answer to such a query would be the i^{th} neighbor of the vertex v. In the *general density model* (also called the *mixed* model) of [43] and [34] no specific representation is assumed. The model allows querying for the degree of a vertex v as well as for the i^{th} neighbor of a vertex v, for any i that is no larger than the degree of v. In both of the above models, the distance function allows edge insertions and deletions whose number is at most a fraction of the number of the edges in the original graph, regardless of the number of vertices. This means that properties such as connectivity are not trivial for testing anymore.

1.3 Massively parameterized property testing

In this thesis we study models of property testing that we call *massively parameterized*. In such models, the tested property is characterized by a complex parameter which is fixed and fully known. For example, Newman [42] studied the property of being accepted by a given bounded-width branching program B. In this case, B is a "massive" parameter given to the testing algorithm in advance, whereas the input to the tester is a word given as an input to B. Hence, the approximation allowed to the tester is only with respect to the input word, while the branching program B is exact. Another example of a massively parameterized testing model is the *general poset domain* model introduced by Fischer et. al [18], who consider monotonicity properties of labelling functions on some partially ordered domain. An important case is where the domain is a directed graph and the labelling function is a vertex coloring. There, the structure of the graph is fixed and given to the algorithms in advance, and the input is its vertex coloring.

The models that we discuss in this thesis can be thought of as variants of the general poset domain. In Part I we consider properties of vertex colorings of trees and ordered forests (in the latter case, there are two orders on the domain). In Part II, we study the *orientation model* introduced by Halevy, Lachish, Newman and Tsur [30]. There, the input is an orientation of the edges in a directed graph whose underlying undirected graph is fixed and known. In all of these models, the distance function depends only on the values of the labelling function on which two given graphs differ, i.e. vertex colors or edge orientations; insertions or deletions of vertices or edges are forbidden.

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. We believe that MPPT is a fertile ground for research, which challenges the development

of new methods and techniques.

Goldreich and Trevisan [27] have proved that any property that is testable in the dense graph model can be tested by uniformly selecting a set of vertices and considering the subgraph that it induces. For massively parameterized models, in contrast, this is generally not the case. Indeed, unlike in many other areas of property testing, many of the algorithms developed for massively parameterized models are non-trivial in construction, and not just in their analysis.

1.4 Overview of the thesis

This thesis is dedicated to the study of various properties in massively parameterized models in which the domain is a (directed or undirected) graph.

In Part I we study properties of vertex colorings in what we call the *vertex coloring model*. The distance between two vertex colorings of the given domain graph is a weighted sum of the vertices for which they differ.

A great deal of Part I is dedicated to variations of the *convexity* property. A vertex coloring is said to be *convex* if it induces connected color components. We study this property for domain graphs that are trees, an important subcase which is motivated by the study of phylogenetic (evolutionary) trees. We show that all the convexity properties that we have considered are testable, providing 1-sided ε-tests, some of which are also non-adaptive. The common feature of all our tests is that they do not necessarily find a direct witness that an input coloring is not convex, but instead they use the knowledge of the domain graph in order to infer the existence of such a witness.

The query complexity of all our tests depends only on the number of colors and on ε. The computational complexity of our tests is polynomial in the size of the graphs, but for most of the tests even this can be avoided if we allow a polynomial time preprocessing stage. On the negative side, we provide a lower bound for the query complexity of testing the convexity property, which applies also for trees which are unweighted paths. The gap between our general upper bound and our lower bound is quadratic. However, we provide a convexity test for paths which is optimal (in its dependence on the number of colors) up to a power of $1/\varepsilon$.

The convexity and convexity variants results are given in Chapters 3 and 4 and are based mainly on [21].

In Chapter 5 we consider more general properties of vertex colorings. The domain

graph that we consider is a rooted ordered forest, namely, a rooted forest in which a linear order is defined on the set of children of every node. We consider testing vertex colorings of such forests for not containing a set of forbidden induced forests. We provide 1-sided tests for two subcases of this property, whose query complexity is independent of the size of the domain forest.

In Part II we consider testing of orientations of an undirected domain graph for being Eulerian. Using the orientation model introduced in [30], the distance between two orientations of the domain graph is the percentage of edges on which they differ.

Although the property of being Eulerian is of a local nature, it appears to be quite involved for testing, since an orientation that is far from being Eulerian might have only a few witnesses of small size. Hence, our results are established through a careful analysis of the characteristics of orientations that are far from being Eulerian. We provide efficient tests for dense graphs and bounded degree graphs. For general graphs, we give a sub-linear algorithm which uses a variation of the expander test as a subroutine. On the negative side, we give super-constant lower bounds for the query-complexity of such tests, which applies also for bounded-degree graphs. Part II is based mainly on [17].

1.5 Global definitions and notations

Graphs. For a graph $G = (V, E)$, we denote the number of vertices in G by $n \stackrel{\text{def}}{=} |V|$ and the number of edges in G by $m \stackrel{\text{def}}{=} |E|$, unless specified otherwise.

Asymptotic notations. Throughout the thesis, we make no attempt to optimize the coefficients of our bounds for the query complexity and computational complexity.

Given a function f on the natural numbers, the notation $\widetilde{O}(f(n))$ is equivalent to $O(f(n) \cdot g(n))$, where g is some function that is polylogarithmic in f. The notation $\text{poly}(f(n))$ denotes any function that is polynomial in f.

Other notations. log denotes the logarithm with base 2, whereas ln denotes the natural logarithm. For any positive integer k, we set $[k] \stackrel{\text{def}}{=} \{1, \ldots, k\}$.

Part I

Testing for Forbidden Posets in Forest Colorings

Chapter 2

Introduction

2.1 Testing for forbidden posets

As said in Chapter 1, combinatorial property testing is concerned with functions from some fixed combinatorial structure S to a finite set of labels. A large class of interesting combinatorial properties are characterized by a set of forbidden induced partially ordered sets, or *posets*. A forbidden poset P is given as a set U of points together with a partial order $\leq_P$ on U and a labelling function $f : U \to L$, where L is a finite set of labels. A set $\mathcal{F}$ of forbidden posets defines a property $\mathcal{P}_\mathcal{F}$ of labelling functions on S, as follows. A labelling function f satisfies $\mathcal{P}_\mathcal{F}$ if it does not induce any forbidden poset $P \in \mathcal{F}$ in S. For example, when testing strings of length n over some finite alphabet Σ, the structure S can be represented as a vector of length n where the input is a labelling function $f : [n] \to \Sigma$. In this case, the set $\mathcal{F}$ of forbidden posets consists of strings over Σ. An input string $w \in \Sigma^*$ satisfies $\mathcal{P}_\mathcal{F}$ if an only if it does not contain any of the strings in $\mathcal{F}$ as a subsequence. Such a property of strings is a special case of a regular language, and as such it is known to be testable [3].

Another classical example of testing for forbidden posets is the well-studied 'monotonicity' property, where the forbidden posets are all the pairs of points $\{x_1, x_2\}$ such that $x_1 \leq_P x_2$ and $f(x_1) > f(x_2)$. A variety of tests, as well as lower bounds, have been devised for monotonicity testing on integer sequences [16, 14], labelled matrices [13], and general poset domains [18]. More general properties defined by forbidden posets have been studied for the matrix domain [19].

2.2 The vertex coloring model

In this part of the thesis we study testing for forbidden posets in several versions of what we call the *vertex coloring model*. This model considers properties of vertex labellings, or *colorings*, of a fixed and known graph, referred to as the *domain graph*. Our testers access the input coloring by querying one vertex at at time, and the distance between two colorings of the domain graph is a weighted sum of the vertices in which they differ. The monotonicity property has already been studied on this model (with a uniform weight function) by Fischer et. al [18], who provided efficient tests for several classes of graphs.

Formally, throughout Part I we assume that $\mu : V \to \mathbb{R}$ is a fixed weight function on the vertices of our domain graph, satisfying $\mu(v) \geq 0$ for every $v \in V$ and $\sum_{v \in V} \mu(v) = 1$. For convenience, define $\mu(U) = \sum_{v \in U} \mu(v)$ for any $U \subseteq V$. The *distance* between two colorings c_1 and c_2 of V is defined as $\mu(\Delta_{c_1,c_2})$, where $\Delta_{c_1,c_2} = \{v \in V \mid c_1(v) \neq c_2(v)\}$. The weight function μ may represent the importance of certain vertices, the cost of modifying them, or the reliability of querying for their color.

In Section 3.1 we consider a stricter model, where the weight function over the vertices is unknown. Such a model is called *distribution-free*, a concept that was introduced in [25] and developed by Halevy and Kushilevitz [29, 28]. 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.

2.3 Convex colorings

Given a graph $G = (V, E)$ and an integer k, a vertex coloring $c : V \to [k]$ 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 (Chapter 3) and several variants of this problem (Chapter 4), on trees. A central motivation for this subcase is the study of phylogenetic (evolutionary) trees, which originated in genetics [40, 51], but appears also in other areas, such as historical linguistics (see [41]). 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 [39] 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.

2.4 Variants of convexity

In Chapter 4 we study several variants of the convexity property defined above, the first of which is *quasi-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). In testing all these relaxed properties we use generalizations of our tests for convexity and quasi-convexity, where a set D of "constraint" vertices is given. A (quasi) k-coloring c is considered *close to a property P under the set D* if it is close to a coloring c' which satisfies P and agrees with c on the values of all the vertices in D.

2.5 Testing for forbidden posets in ordered rooted forests

In Chapter 5 we consider testing vertex colorings of a fixed ordered rooted forest.

A *rooted forest* is a collection of rooted trees. A root of a tree in a rooted forest is referred to as *a root in the forest*. An *ordered rooted forest* is a rooted forest in which a linear order is defined on the set of children of every node, as well as on the set of trees in the forest. Therefore, an ordered rooted forest induces two partial orders on its nodes: The familiar "ancestry" relation, in which the comparable node pairs are those lying on a simple path to the root; and a "left-to-right" order, in which the comparable pairs are those which are incomparable with respect to ancestry. For two vertices u and v, we say that *u is left of v* if any of the following applies: 1. u and v are children of the same node w and u precedes v in the order defined on w's children; 2. $u \in T_1$ and $v \in T_2$, where T_1 and T_2 are two trees such that T_1 precedes T_2 in the order defined on trees in the forest; 3. There exist two vertices a, b such that u is a descendant of a, v is a descendant of b and a is left of b.

When considering vertex colorings of ordered rooted forests, a forbidden poset is a colored rooted ordered forest, specifying a forbidden setting of ancestry and left-to-right relation pairs over a set of colored vertices. We examine two subcases of this property. In the first case, the forbidden set consists of (ancestral) chains, and in the second case, it consists of one general ordered rooted forest.

We note that properties of ordered labelled trees have been studied by [37]. However, they do not assume knowledge of the tree structure, while on the other hand, their distance function allows moves of an entire subtree, insertions and deletions, while we only allow recoloring vertices.

2.6 Our results

Testing convexity in trees (Chapter 3). We show that convexity of tree colorings is testable, providing a 1-sided, non-adaptive, distribution-free ε-test for every $\varepsilon > 0$. The query complexity of our test for k-colorings is $O(k/\varepsilon)$, 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 $O(\sqrt{k}/\varepsilon^3)$ and additional time complexity $\widetilde{O}(\sqrt{k}/\varepsilon^3)$.

On the negative side, we prove a lower bound of $\Omega(\sqrt{k}/\sqrt{\varepsilon})$ on the query complexity of testing convexity of paths even in the unweighted model.

Variants of convexity in trees (Chapter 4). We show that quasi-convexity of tree colorings is testable, giving a 1-sided, non-adaptive (but not distribution-free) ε-test for every $\varepsilon > 0$. The query complexity of our tests for quasi k-colorings is $O(k/\varepsilon^2)$ and the additional time complexity $O(kn/\varepsilon)$. 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 $O(n^2)$.

As for testing for convexity and quasi-convexity under a set D of constraint vertices, we show that for both properties, it is enough to augment the query set of our regular tests with the constraint vertices, and therefore, the addition to the query and computational complexity is $O(|D|)$.

Finally, we provide 1-sided ε-tests for every $\varepsilon > 0$ for the relaxed convexity properties. For ℓ-convexity we give a test with query complexity $\widetilde{O}(\ell/\varepsilon)$ and time complexity $O(\ell n)$. For ℓ-quasi-convexity we provide a test with query complexity $\widetilde{O}(\ell/\varepsilon^2)$ 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 $\widetilde{O}(\ell/\varepsilon)$ 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 $\widetilde{O}(\ell/\varepsilon^2)$ and computational complexity $O(\ell n)$.

Forbidden posets in ordered forests (Chapter 5). We provide 1-sided, non-adaptive ε-tests for every $\varepsilon > 0$ for both subcases that we have studied. For the case where the set of forbidden posets consists of (ancestral) chains, the query complexity

is poly $\left(\frac{1}{\varepsilon}\right)$ and the additional time complexity is $O(n)$, in a preprocessing stage. For the case where the forbidden set consists of one general ordered forest, the query and time complexity are both poly $\left(\frac{1}{\varepsilon}\right)$. Moreover, our complexity bounds for both cases are independent of the number of colors k.

Chapter 3

Testing Convexity in Trees

Throughout this chapter, our domain graph is a fixed tree $T = (V, E)$ and our input is a k-coloring $c : V \to [k]$ of T. 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.

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*.

Finally, for any two distinct vertices u and 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.

3.1 A distribution-free convexity test for trees

Below we provide a simple test for convexity, which samples vertices according to the weight function μ on V and then queries their colors. Our test is distribution-free (see Section 2.2), as it uses the distribution μ as a black-box only. We note that the standard definition of distribution-free testing allows in addition queries for the color of determined vertices, but our test will do better and use only sample vertices. To reject the input, the sample does not necessarily need to contain a forbidden subpath. Instead, the algorithm uses the information supplied by the queried vertices, together with the knowledge of the structure of the tree T, to infer the existence of a forbidden subpath. The main idea behind the algorithm

is that if a coloring is ε-far from being convex, then, with high probability, either a forbidden subpath is sampled or there exists a vertex which is a "crossroad" of sampled subpaths with conflicting colors.

Algorithm 3.1.1.

1. *Query* $\left\lceil \frac{8k \ln 12}{\varepsilon} \right\rceil$ *vertices, where each vertex is independently chosen according to the distribution* μ*. Let* X *denote the sample.*

2. *If* X *includes a forbidden subpath, reject.*

3. *Otherwise, if there exists* $w \in V$ *such that any value of* $c(w)$ *implies a forbidden subpath, reject. In other words, reject if there exist* $w \in V$ *and* $u_1, u_2, v_1, v_2 \in X$ *such that* $c(u_1) = c(u_2) \neq c(v_1) = c(v_2)$*, and* w *belongs to both the path between* u_1 *and* u_2 *and the path between* v_1 *and* v_2*.*

4. *Otherwise, accept.*

Theorem 3.1.2. *For every* $\varepsilon > 0$*, Algorithm 3.1.1 is a 1-sided* ε*-test for convexity of* k*-colorings of trees. The query complexity of the test is* $O(k/\varepsilon)$ *and the time complexity is* $O(n)$*. This can be implemented in running time* $\widetilde{O}(|X|) = \widetilde{O}(k/\varepsilon)$ *using a preprocessing stage of time* $O(n)$*.*

It is easy to see that the query complexity is as stated. We show how to implement the stated computational steps under the time complexity requirements in Subsection 3.1.1. Clearly, a convex coloring is always accepted by Algorithm 3.1.1, as it does not contain forbidden subpaths. Thus, it remains to show that every coloring which is ε-far from being convex is rejected with probability at least $\frac{2}{3}$.

A color $i \in [k]$ is called *abundant* if $\mu_i(V) \geq \varepsilon/2k$. 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 $\varepsilon/8k$. We say that a vertex v is *heavy* if $\mu(v) \geq \varepsilon/8k$.

Lemma 3.1.3. *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 $\varepsilon/4k$ (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 $\varepsilon/4k$, 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 $\varepsilon/4k$. Due to the minimality of $C_u^{(v)}$, we must have $w = u$. Thus $C_v^{(u)}$ is of i-weight larger than $\varepsilon/4k$, and, since there are no heavy i-vertices, the i-weight of $C_v^{(u)} \setminus \{u\}$ is at least $\varepsilon/8k$. Therefore, both $C_u^{(v)}$ and $V \setminus \{C_u^{(v)} \cup \{u\}\}$ have i-weight of at least $\varepsilon/8k$, 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.

Lemma 3.1.4. *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 $\mu_i(C_w^{(u)}) \geq \varepsilon/8k$. 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 $\varepsilon/8k$. One of these sets does not contain $C_u^{(w)}$. Thus, $\mu_i(C_w^{(u)}) \geq \varepsilon/8k$. Similarly, it follows that $\mu_i(C_w^{(v)}) \geq \varepsilon/8k$, and hence w is i-balanced. □

Proposition 3.1.5. *For every k-coloring c of T that is ε-far from being convex, there exist two different abundant colors i and j such that $B_i \cap B_j \neq \emptyset$.*

Proof. Note first that there must be at least two abundant colors. Otherwise, the total weight of the vertices of non-abundant colors is smaller than ε, and we may obtain a convex coloring of T by recoloring all of them with the only abundant color.

Suppose that the connected sets B_i are all disjoint. We show a convex k-coloring c' of T that is ε-close to c, which leads to a contradiction. Define c' as follows. For every vertex v and abundant color i, let $d(v, B_i)$ denote the "walking" distance on T between v and B_i, i.e. the length of the path from v to (the connected) B_i. Color every vertex v with i such that $d(v, B_i)$ is minimal, choosing the minimal index i in case of a tie. In particular, we color all the vertices in B_i with i.

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 $B_i \subseteq C_w^{(u)}$. By

Lemma 3.1.4, B_i is connected. By the definition of c', $w \notin B_i$, and thus either $B_i \subseteq C_u^{(w)}$ or $B_i \in C_w^{(u)}$. Assume that $B_i \subseteq C_u^{(w)}$. Then $d(u, B_i) = d(w, B_i) + 1$ and $d(u, B_j) \leq d(w, B_j) + 1$. But from the way u and w are colored by c' we have $d(u, B_i) - d(u, B_j) < d(w, B_i) - d(w, B_j)$, a contradiction. Hence, $B_i \subseteq C_w^{(u)}$.

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 $B_i \subseteq C_{w_1}^{(u)}$ and $B_i \subseteq C_{w_2}^{(v)}$, a contradiction.

We now show that c' is ε-close to c. Consider a vertex w whose color has been changed from one abundant color i into another (abundant) color j. Surely $w \notin B_i$. Furthermore, either B_j is on the path between B_i and w or w is on the path between B_i and B_j. Consider the edge (u, v) on the path between B_i and B_j where $u \in B_i$ and $v \notin B_i$. We call (u, v) the *ij-bridge*. Then $w \in C_u^{(v)}$. Now, let (u, v) be the ij-bridge where i and j are some two distinct abundant colors. Suppose that $\mu_i(C_u^{(v)}) \geq \varepsilon/4k$. By the definition of B_i, v is not a heavy i-vertex, and thus $\mu_i(C_u^{(v)} \setminus \{v\}) > \varepsilon/8k$. Since v is not i-balanced, we have $\mu_i(C_v^{(u)}) < \varepsilon/8k$, but this is impossible, as u is i-balanced or heavy. We thus conclude that $\mu_i(C_u^{(v)}) < \varepsilon/4k$ for every ij-bridge (u, v).

Let T' be the tree created from T by contracting every set B_i into a single vertex and removing all 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 $\ell \leq k$.

Claim 3.1.6. $\sum_{i=i}^{\ell} D_i \leq 2(\ell - 1) \leq 2(k - 1)$.

Proof. By the definition of T', all its leaves are B_i's. We now prove the claim 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 $\ell > 2$. 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, $\sum_{i=i}^{\ell} D_i \leq 2(\ell-2)+2 = 2(\ell-1)$. $\square$

From the discussion above, it follows that the total weight of recolored vertices among those whose original color was abundant is less than $\sum_{i=i}^{\ell} D_i(\varepsilon/4k) \leq 2(k-1)(\varepsilon/4k) < \frac{\varepsilon}{2}$. As non-abundant colors are of weight smaller than $\varepsilon/2k$ each, their total weight is smaller than $\varepsilon/2$. Thus, c' is ε-close to c. $\square$

Proof of Theorem 3.1.2. We have shown that for every k-coloring c that is ε-far from being convex, there exist i, j and a vertex w such that $w \in B_i \cap B_j$. Clearly w must be i-balanced or j-balanced or both. Suppose that w is not balanced with respect to one of the colors, say i. Then w must be a heavy i-vertex and j-balanced. In such a case $\mu(w) \geq \varepsilon/8k$ and there exist two disjoint sets $W_1^j, W_2^j \subseteq V_j$, each of weight at least $\varepsilon/8k$, such that 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 X contains w and at least one vertex from each of the sets W_1^j and W_2^j, then Algorithm 3.1.1 rejects the input in Step 2. Now, the probability for any of the sets W_1^j and W_2^j or of w to not intersect the sample set X, is at most $(1-\varepsilon/8k)^{\frac{8k\ln 12}{\varepsilon}}$. Thus, by the union bound, the algorithm will fail with probability at most $3(1-\varepsilon/8k)^{\frac{8k\ln 12}{\varepsilon}} < 3\exp(-\ln 12) = 1/4$.

Otherwise, if w is both i-balanced and j-balanced, then there exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\varepsilon/8k$, and two disjoint sets $W_1^j, W_2^j \subseteq V_j$, each of weight at least $\varepsilon/8k$, 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 X contains at least one vertex from each of the sets $W_1^i, W_2^i, W_1^j, W_2^j$, Algorithm 3.1.1 rejects the input in Step 3. The probability for any given set of the above to not intersect X is at most $(1-\varepsilon/8k)^{\frac{8k\ln 12}{\varepsilon}}$. Thus, by the union bound, the algorithm will fail with probability at most $4(1-\varepsilon/8k)^{8k\ln 12/\varepsilon} < 4\exp(-\ln 12) = 1/3$. $\square$

3.1.1 Implementing the computation step in Algorithm 3.1.1

We now specify a procedure implementing Steps 2 and 3 of Algorithm 3.1.1 in time $O(n)$, where the constants are independent of k and ε. Later we show how this procedure can be executed in time $\widetilde{O}(|X|) = \widetilde{O}(k/\varepsilon)$ 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 sample 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. [35]). 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 3.1.7. *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 3.1.7 implements Steps 2 and 3 of Algorithm 3.1 correctly.

Lemma 3.1.8. *For every vertex v the following holds:*

1. *If Procedure 3.1.7 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 3.1.7 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 3.1.7 is correct, as it rejects any sample X containing or implying a forbidden subpath, and accepts otherwise.

Note that Procedure 3.1.7 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 3.1.9. *$\widehat{X}$ 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 [32], [48]). After yielding the set X, we build the tree T_X using the following procedure.

Procedure 3.1.10.

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 3.1.7 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 3.1.11. *Procedure 3.1.10 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 3.1.10 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 3.1.10. $\square$

3.1.2 Testing convexity with constraints

Given a graph $G = (V, E)$, a set $D \subseteq V$ of "constraints", a weight function $\mu : V \setminus D \to \mathbb{R}$, and a k-coloring $c : V \to [k]$, we say that c is *ε-close to convexity under D* if there exists a convex k-coloring $c' : V \to [k]$ which agrees with c on the values of the vertices in D and whose restriction to $V \setminus D$ is ε-close to that of c. If c is not ε-close to convexity under D, then we say that c is *ε-far from convexity under D*.

We now show that for a domain tree $T = (V, E)$, it is enough to test for convexity under a set of constraints D by simply adding D to the query set and then searching for a forbidden subpath as in Algorithm 3.1.1.

Algorithm 3.1.12. *Identical to Algorithm 3.1.1, except that in the end of Step 1 we also query all the constraint vertices in D.*

Note that Algorithm 3.1.12 still follows the common definition of being distribution-free.

Proposition 3.1.13. *Algorithm 3.1.12 is a 1-sided ε-test for convexity of k-colorings of trees under a set of constraints D. The query complexity of the test is $O(k/\varepsilon + |D|)$ and the time complexity is $O(n)$. This can be implemented in time $\widetilde{O}((k/\varepsilon + |D|))$ using a preprocessing stage of $O(n)$.*

Proof. Clearly, the computational complexity corresponds to that of Algorithm 3.1.1 (see Section 3.1.1). If we use preprocessing, the complexity now depends on the size of the query set plus the constraint set, whereas, without preprocessing, the complexity depends only on the number of vertices in the tree. Hence, the computational complexity demands are satisfied. Clearly, Algorithm 3.1.12 always accepts a convex k-coloring. We now show that every k-coloring which is ε-far from convexity under D is rejected by the algorithm with probability at least $2/3$.

Define the sets B_i of i-balanced vertices and heavy i-vertices as in the proof of Theorem 3.1.2. Note that the weight function μ is defined only on $V \setminus D$. We have seen that for every i-balanced vertex w, with high probability we sample two i-vertices u and v such that w is on the path between u and v. Therefore, if the B_i's are not disjoint, then Algorithm 3.1.1 rejects with high probability, as shown in its proof of correctness. In addition, if there exists an i-vertex $w \in D$ such that $w \in B_j$ for $i \neq j$, then, clearly, with high probability, Algorithm 3.1.1 finds a forbidden subpath and rejects.

Assume now that the B_i's are disjoint, and that every constraint vertex $w \in D \cap B_i$ is an i-vertex. Consider the tree T' created from T by contracting every set B_i into a single vertex and removing all vertices which do not belong to a path between B_i's and/or constraint vertices. We refer to the B_i's and the constraint vertices as *special vertices* in T'. Suppose that T' contains either (a) a forbidden subpath comprised of special vertices; or (b) a "critical" vertex w and special vertices u_1, u_2, v_1, v_2 such that $c(u_1) = c(u_2)$, $c(v_1) = c(v_2)$ and w is both on the path between u_1 and u_2 and on the path between v_1 and v_2. One can prove that Algorithm 3.1.1 rejects with probability at least 2/3 in either case, using the same techniques as in the proof of Theorem 3.1.2. Note that the constraint vertices are sampled by the algorithm with probability 1.

On the other hand, suppose that T' does not contain a forbidden subpath of special vertices or a critical vertex as above. Then we may enlarge the B_i's by adding to every B_i all the i-colored constraint vertices and the vertices on the paths to them. For constraint vertices colored with a non-abundant color i, we create a new set B_i containing all the i-colored constraint vertices and the paths between them. We now have at most one set B_i for every color i, and all the sets are disjoint. Thus, we can define a convex coloring c' of T in the same way as was done in the proof of Proposition 3.1.5.

To prove that the restriction of c' to $V \setminus D$ is ε-close to that of c, we again use the fact that for every B_i, we have only recolored i-vertices which are in $C_u^{(v)}$, where (u, v) is an ij-bridge (as defined in the proof of Proposition 3.1.5). For abundant colors i, we have shown that for an ij-bridge (u, v) we have $\mu_i(C_u^{(v)}) < \varepsilon/4k$. Note that in the proof we have only used the fact that each B_i contains all the i-balanced vertices and the heavy i-vertices, and therefore we may also apply it here, regardless of whether j is an abundant color or not. Hence, we obtain the same upper bounds for the weight of recolored vertices of both abundant and non abundant colors, and thus, c' is ε-close to c on $V \setminus D$. We conclude that if c is ε-far from convexity under D then Algorithm 3.1.1 rejects with probability at least 2/3. □

3.2 A lower bound for testing convexity on trees

Theorem 3.2.1. *For every $0 < \varepsilon < 1/8$, every (adaptive) ε-test for convexity of k-colorings of trees must use more than $\sqrt{3\frac{(k-1)}{64\varepsilon}}$ queries in the worst case. This is specifically true for the case where T is a path and μ is a uniform distribution.*

Proof. Let T be a path of length n. According to Yao's theorem [53], 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 $q \leq \sqrt{\frac{3(k-1)}{64\varepsilon}}$ 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 $q \leq \sqrt{\frac{3(k-1)}{64\varepsilon}}$ 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 3.2.2. *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 3.2.3. *Let $\widetilde{D_N}$ be the distribution of k-colorings selected by uniformly choosing $(1-8\varepsilon)k$ colors to appear in one interval and $4\varepsilon k$ colors to appear in two intervals. The placements of the colors are then chosen uniformly at random.*

Definition 3.2.4. *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 [16] for details.

Lemma 3.2.5. *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 $2\varepsilon k$ colors that appear twice must appear on adjacent intervals. Otherwise, there are more than $2\varepsilon k$ 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 $2\varepsilon k$ 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

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

for choosing the $2\varepsilon k$ 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

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

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 3.2.6. *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,

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

Since $D_{\widetilde{N}}$ is a uniform distribution over C and D_N is a uniform distribution over C_ε, and, by Lemma 3.2.5, $|C_\varepsilon| > \frac{3}{4}|C|$, for every coloring $c \in C_\varepsilon$ we have $\Pr_N[c] < \frac{4}{3} \Pr_{\widetilde{N}}[c]$. Therefore,

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

□

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 3.2.7. *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 $4\varepsilon kq(q-1)\binom{k-2}{q-2}(q-2)!$, as there are $4\varepsilon k$ 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

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

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

From Observation 3.2.6, 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 3.2.8. *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 3.2.7. □

We now complete the proof of Theorem 3.2.1 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 3.2.8, $0 \leq \Pr_{\mathcal{A},P}[g] - \Pr_{\mathcal{A},N}[g] < \frac{1}{3}\Pr_{\mathcal{A},N}[g]$ for any colorful answer sequence. Thus, $0 \leq a_P - a_N^c < \frac{1}{3}$. In addition, from Lemma 3.2.7, $0 \leq a_N^d \leq \alpha_N < \frac{1}{4}$. Hence

$|a_P - a_N| < \frac{1}{3}$. $\square$

3.3 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 3.1.1 when the number of colors k is large enough with respect to $1/\varepsilon^3$. 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 [3]. However, the query complexity obtained there, though polynomial in $\frac{1}{\varepsilon}$, 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 Section 3.2, our algorithm is optimal up to a power of $\frac{1}{\varepsilon}$.

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 3.3.1.

1. *Query $q \geq \frac{1280\sqrt{k}}{\varepsilon^2}$ 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 $z \geq \frac{5}{\varepsilon} \ln 12$ 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 3.3.1 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 3.3.1 does not generate 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 3.3.2. *For every $\varepsilon > 0$, Algorithm 3.3.1 is a 1-sided ε-test for convexity of k-colorings of paths, with query complexity $O(\sqrt{k}/\varepsilon^3)$. The additional time complexity is $\widetilde{O}(\sqrt{k}/\varepsilon^3)$ 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 $\widetilde{O}(\sqrt{k}/\varepsilon^3)$ 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 $\frac{2}{3}$.

For every color i such that $\mu_i(V) > 0$, let l_i be the leftmost vertex such that $\mu_i([v_L, l_i]) \geq \varepsilon\mu_i(V)/4$. 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 $\mu_i([r_i, v_R]) \geq \varepsilon\mu_i(V)/4$. 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 $\varepsilon\mu_i(V)/4$. 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 $\mu_i(M_i) > (1 - \varepsilon/2)\mu_i(V)$.

We say that a color $i \in [k]$ is *abundant* if $\mu_i(V) \geq \varepsilon/8k$, 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 $\Sigma_{i \in \mathcal{N}_\mathcal{A}} \mu_i(V) < \varepsilon/8$. 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 3.3.3. *If c is ε-far from being convex then $\sum_{i \in \mathcal{A}_{bad}} \mu_i(V) \geq \frac{\varepsilon}{8}$.*

Proof. Assume on the contrary that $\sum_{i \in \mathcal{A}_{bad}} \mu_i(V) < \frac{\varepsilon}{8}$. 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 $\varepsilon\mu_i(V)/4$. Hence, the non-i-weight of every interval colored with i in Phase 1 is smaller than $\varepsilon\mu_i(V)/4$, and thus the total weight of vertices colored in Phase 1 differently than c is less than $\varepsilon/4$. On the other hand, since for every good color i we have $\mu_i(M_i) > (1-\varepsilon/2)\mu_i(V)$, the total weight of vertices colored in Phase 1 is

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

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

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

Lemma 3.3.4. *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 3.3.1 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 3.3.3 we have $\sum_{i\in\mathcal{A}_{bad}} \mu_i(V) \geq \frac{\varepsilon}{8} > 0$. Let i be a color in $\mathcal{A}_{bad}$. By the definition of L_i, we have $\mu_i(L_i) \geq \varepsilon\mu_i(V)/4$, and since i is abundant, we have $\mu_i(L_i) \geq \varepsilon^2/32k > \varepsilon^2/512$. Thus, the probability of not choosing an i-vertex in L_i in step 1 is smaller than

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

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 $\mu_i([0, \tau_i]) = \varepsilon\mu_i(V)/4$, and denote $\mathcal{L}_i = [0, \tau_i]$. Let η_i be the maximum point in S such that $\mu_i([\eta_i, 1)) = \varepsilon\mu_i(V)/4$, and denote $\mathcal{R}_i = [\eta_i, 1)$. Clearly, $\mathcal{L}_i$ and $\mathcal{R}_i$ are disjoint intervals. Moreover, note that sampling a point in $\mathcal{L}_i$ 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 $x \in \mathcal{L}_i$ 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 $\mathcal{L}_i$. Clearly, $\mu(I_i) = \mu_i(\mathcal{L}_i) = \varepsilon\mu_i(V)/4$. We now define $\lfloor \frac{64\mu_i(\mathcal{L}_i)k}{\varepsilon^2} \rfloor$ disjoint subsets of I_i of measure $\frac{\varepsilon^2}{64k}$ 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 $|X_1| \geq 256\sqrt{k}/\varepsilon^2$ and $|X_2| \geq 1024\sqrt{k}/\varepsilon^2$. We next prove that, with high probability, X_1 contains samples from at least $\sqrt{k}$ distinct subcolor sets in $\mathcal{L}_i$. For all colors $i \in \mathcal{A}_{bad}$ we have $\mu_i(\mathcal{L}_i) = \frac{\varepsilon}{4}\mu_i(V)$, and since these colors are abundant, we have $\mu_i(\mathcal{L}_i) \geq \frac{\varepsilon^2}{32k}$. From Lemma 3.3.3 we have $\sum_{i \in \mathcal{A}_{bad}} \mu_i(V) \geq \frac{\varepsilon}{8}$, and thus $\sum_{i \in \mathcal{A}_{bad}} \mu_i(\mathcal{L}_i) \geq \frac{\varepsilon^2}{32}$. 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 $\frac{\varepsilon^2}{64k}$. Hence, the total measure of subcolor sets is greater than $\frac{\varepsilon^2}{32} - k\frac{\varepsilon^2}{64k} = \frac{\varepsilon^2}{64}$, 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 $\mathcal{L}_i$'s, and therefore

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

where the last inequality follows from Lemma 3.3.3.

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 $\mathcal{L}_i$. Since the measure of every subcolor set is $\frac{\varepsilon^2}{64k}$, we have $\Sigma_{i \in \mathcal{C}} \mu_i(\mathcal{L}_i) \geq \frac{\varepsilon^2}{64\sqrt{k}}$. By definition, it follows that $\Sigma_{i \in \mathcal{C}} \mu_i(\mathcal{R}_i) \geq \frac{\varepsilon^2}{64\sqrt{k}}$. Hence, the probability that X_2 does not contain any i-point for some $i \in \mathcal{C}$ is at most

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

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

Proof of Theorem 3.3.2. Assume that the input coloring c is ε-far from being convex. According to Lemma 3.3.4, 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 $\varepsilon\mu_i(V)/4$ (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 $\varepsilon/(4(1+\varepsilon/4)) \geq \varepsilon/5$. 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 $\varepsilon/5$. Since in Step 2 we query $z \geq \frac{5}{\varepsilon}\ln 12$ 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 $1/4 + 1/12 = 1/3$. $\square$

Chapter 4

Variants of The Convexity Property

Throughout the chapter, our domain is a fixed tree $T = (V, E)$, and our input is either a k-coloring or a quasi k-coloring, denoted by c in both cases. As in Chapter 3, for every color i, we define V_i to be the set of vertices v in V such that $c(v) = i$. *i-vertices*, *non-i-vertices*,*i-weight* ,*non-i-weight* and the components $C_u^{(v)}$ are defined as in Chapter 3.

4.1 A quasi-convexity test for trees

Given a quasi k-coloring $c : V \to \{0, 1, \ldots, k\}$ of the domain tree T, we say that a vertex $c \in V$ 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.

Algorithm 4.1.1.

1. *Query* $\lceil 48k/\varepsilon \rceil$ *vertices, where each vertex is independently chosen according to the distribution defined by* μ. *Let* X *denote the sample.*

2. *If* X *includes a forbidden subpath, reject.*

3. *Otherwise, if there exists* $w \in V$ *such that any value of* $c(w)$ *implies a forbidden subpath, reject. In other words, reject if there exist* $w \in V$ *and* $u_1, u_2, v_1, v_2 \in X$

such that $c(u_1), c(u_2), c(v_1), c(v_2) > 0$ *and* $c(u_1) = c(u_2) \neq c(v_1) = c(v_2)$*, where* w *belongs both to the path between* u_1 *and* u_2 *and to the path between* v_1 *and* v_2*.*

4. *Otherwise, repeat the following* $\lceil 16/\varepsilon \rceil$ *times independently:*

   - *Choose a vertex* w *according to the distribution defined by* μ *and query it. If* w *is colored, do nothing.*
   - *Otherwise, if* w *is uncolored, define a subtree* T_w^i *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 *does not contain a forbidden subpath). Now denote* $T_w^i \stackrel{\text{def}}{=} C_{v_i}^{(w)}$ *for every such* v_i*. Query* $\lceil \log_{1/(1-\varepsilon/8)} 8 \rceil$ *vertices in each* T_w^i*, where each vertex is independently chosen according to the distribution defined by* μ *conditioned on* T_w^i*.*

5. *If the union of* X *and vertices queried in Step 4 includes a forbidden subpath, then reject. Otherwise, accept.*

Theorem 4.1.2. *For every* $\varepsilon > 0$*, Algorithm 4.1.1 is a 1-sided* ε*-test for quasi-convexity of quasi* k*-colorings of trees, with query complexity* $O(k/\varepsilon^2)$*. The time complexity of the test is* $O(kn/\varepsilon)$*, and can be implemented in time* $\widetilde{O}(k/\varepsilon^2)$ *with a preprocessing stage of time* $O(n^2)$*.*

Note that for ε small enough, we have $\log_{1/(1-\varepsilon/8)} 8 < \frac{\ln 8}{\varepsilon/16}$. Thus, it is easy to see that the query complexity is as stated. In Section 4.1.1 we show how to detect forbidden subpaths, as done in Steps 2,3 and 5 of the Algorithm, in time $O(n)$, or $\widetilde{O}(k/\varepsilon^2)$ with a preprocessing stage of time $O(n)$. In Step 4, 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, if c is quasi-convex then it is always accepted by Algorithm 4.1.1, since it does not contain any forbidden subpaths. Therefore, it remains to show that every

quasi k-coloring c that is ε-far from being quasi-convex is rejected with probability at least $\frac{2}{3}$.

Here we say that a color $i \in [k]$ is *abundant* if $\mu(V_i) \geq \varepsilon/4k$. 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 $\varepsilon/16k$. We say that a vertex v is *heavy* if $\mu(v) \geq \varepsilon/16k$.

For every abundant color i, we define the set B_i as the union of i-balanced vertices and heavy i-vertices. Assume, without loss of generality, that the abundant colors are numbered $i = 1, \ldots, \ell$. The proof of the following lemma is identical to the proofs of Lemma 3.1.3 and Lemma 3.1.4.

Lemma 4.1.3. *B_i is non-empty and connected for every abundant color i.*

Observation 4.1.4. *If $\ell = 0$ then c is ε-close to being quasi-convex.*

Proof. Clearly, in such a case, the total weight of colored vertices is at most $\varepsilon/4$, and hence we may obtain an $\varepsilon/4$-close quasi-convex coloring by setting all colored vertices to be uncolored. □

Suppose now that $\ell > 0$ and the B_i's are all disjoint. We say that a vertex v is an *outsider* if it does not belong to any B_i or to a path between two B_i's, and, moreover, none of the vertices on the path between v and its closest B_i belongs to a path between two B_i's. For every abundant color i, let M_i be the union of B_i and of all the outsider vertices whose closest B_j is B_i. Clearly, all the M_i's are connected and disjoint. The proof for the following lemma is based on ideas similar to those used in the proof of Proposition 3.1.5.

Lemma 4.1.5. $\sum_{i=1}^{\ell} \mu_i(V \setminus M_i) \leq \varepsilon/4$.

Proof. First note that if $\ell = 1$ then for the only abundant color we have $V \setminus M_1 = \emptyset$, so the lemma is trivially true. We thus may assume that $\ell > 1$. As in the proof of Proposition 3.1.5, 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 in which $u \in B_i$ and $v \notin B_i$ (clearly, $v \notin M_i$). Observe that, for every abundant color i, the set $V \setminus M_i$ is the union of the components $C_u^{(v)}$ for all ij-bridges (u,v).

Suppose that $\mu_i(C_u^{(v)}) \geq \varepsilon/8k$ for some abundant color i and some ij-bridge (u,v). As $v \notin B_i$, v is not a heavy i-vertex, and thus $\mu_i[C_u^{(v)} \setminus \{v\}] > \varepsilon/16k$. Since v is not i-balanced, we have $\mu_i[C_v^{(u)}] < \varepsilon/16k$, but this is impossible, as u is i-balanced or a heavy i-vertex. We conclude that $\mu_i(C_u^{(v)}) < \varepsilon/8k$ for every ij-bridge (u,v).

Now, let T' be the tree created from T by contracting every set M_i into a single vertex. Let D_i be the degree of M_i in T'. Note that by the definition of T', all its leaves are M_i's. Clearly, for every abundant color i, D_i is the number of ij-bridges. Similarly to Claim 3.1.6, we have that

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

From the discussion above,

$$\sum_{i=1}^{\ell} \mu_i(V \setminus M_i) = \sum_{i=1}^{\ell} \sum_{\substack{(u,v) \text{ is an} \\ ij-\text{bridge}}} \mu_i(C_u^{(v)}) < \sum_{i=1}^{\ell} D_i(\varepsilon/8k) < \frac{\varepsilon}{4}.$$

□

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 B_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 an *i-satellite*, for some abundant color i, if it is an i-vertex associated with i. We call a vertex a *satellite* if it is an i-satellite for some abundant color i.

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. If F can be made monotone by changing the color of satellite and uncolored vertices of weight at most $\varepsilon/4$, and any amount of other vertices in F, then we say that F is *good*. Otherwise, we say that F is *bad*.

We call an uncolored vertex w in F *good* if $\mu_i(T_w) \leq \frac{\varepsilon}{8}\mu(T_w)$, where T_w is the subtree of F rooted in w and i is the abundant color associated with w. Otherwise, we say that such an uncolored w is *bad*.

Lemma 4.1.6. *If the total weight of bad vertices in F is at most $\varepsilon/8$, then F is good.*

Proof. The proof is very similar to the one presented in [18] for the test of monotonicity over rooted trees. Define a monotone coloring of F as follows. Let U be the set of all good uncolored vertices in F. Let U_r be the set of topmost vertices in U. Set all the descendants of vertices in U_r to be uncolored, and color the rest

of the vertices in F with the color associated with them. Clearly, in the obtained coloring, 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 that we have colored are bad ones, whose weight is at most $\varepsilon/8$. As for satellite vertices, we have only changed the color of ones within subtrees of good uncolored vertices whose roots are in U_r. Since these subtrees are disjoint, and the weight of satellite vertices is a fraction of at most $\varepsilon/8$ of the weight of any good subtree, the total weight of satellite vertices thus changed is at most $\varepsilon/8$. Hence, we have changed a weight of at most $\varepsilon/4$ uncolored and satellite vertices, and therefore F is good. □

Lemma 4.1.7. *If $\ell > 0$ and all of the following conditions apply:*

(a) *The B_i's are all disjoint;*

(b) *The total weight of uncolored vertices inside B_i's is at most $\varepsilon/4$;*

(c) *F is good;*

then the input coloring c is ε-close to being quasi-convex.

Proof. We show that in such a case, there exists a quasi-convex coloring c' of T that is ε-close to c. Define c' as follows. For every abundant color i, color all the vertices of B_i with i. Then choose a monotone coloring for F which changes a minimum weight of uncolored and satellite vertices. Finally, set the rest of the vertices to be uncolored. One can see that c' is quasi-convex.

We now show that c' is ε-close to c. First, consider vertices of abundant colors, excluding satellites, whose color has been changed. From Lemma 4.1.5, the total weight of such vertices is at most $\varepsilon/4$. Now consider uncolored vertices inside B_i's. From Condition (b), the total weight of such vertices is at most $\varepsilon/4$. As for satellites and uncolored vertices in F, from Condition (c), the total weight of these is at most $\varepsilon/4$. Finally, for non-abundant colors, since they are of weight smaller than $\varepsilon/4k$ each, their total weight is smaller than $\varepsilon/4$. We conclude that c' is ε-close to c. □

To complete the proof of Theorem 4.1.2, we need the next three lemmas.

Lemma 4.1.8. *If $\ell > 0$ and the B_i's are not disjoint, then the input is rejected by Step 2 or Step 3 of Algorithm 4.1.1 with probability at least $2/3$.*

Proof. The proof is similar to the main proof of Theorem 3.1.2. Let w be a vertex such that $w \in B_i \cap B_j$ for $i \neq j$. Clearly, w must be i-balanced or j-balanced or both. Suppose that w is not balanced with respect to one of the colors, say, i. Then w must be a heavy i-vertex and j-balanced. In such a case $\mu(w) \geq \varepsilon/16k$ and there exist two disjoint sets $W_1^j, W_2^j \subseteq V_j$, each of weight at least $\varepsilon/16k$, such that every path between vertices $v_1 \in W_1^j$ and $v_2 \in W_2^j$ passes through w. Hence, if the sample X contains w and at least one vertex from each of the sets W_1^j and W_2^j, then Algorithm 4.1.1 rejects the input in Step 2. Now, the probability for each of W_1^j, W_2^j, or w to not intersect X is at most $(1 - \varepsilon/16k)^{\frac{48k}{\varepsilon}} < \exp(-3)$. By the union bound, the algorithm will fail with probability at most $3\exp(-3) \leq 1/4$.

Otherwise, if w is both i-balanced and j-balanced, then there exist two disjoint sets $W_1^i, W_2^i \subseteq V_i$, each of weight at least $\varepsilon/16k$, and two disjoint sets $W_1^j, W_2^j \subseteq V_j$, each of weight at least $\varepsilon/16k$, where every path between vertices $u_1 \in W_1^i$ and $u_2 \in W_2^i$ passes through w and every path between vertices $v_1 \in W_1^j$ and $v_2 \in W_2^j$ passes through w. Therefore, if the sample X contains at least one vertex from each of the sets $W_1^i, W_2^i, W_1^j, W_2^j$, then Algorithm 4.1.1 rejects the input in Step 3. Now, the probability for any specific set of the above to not intersect X is at most $(1 - \varepsilon/16k)^{\frac{48k}{\varepsilon}} = \exp(-3)$. Thus, by the union bound, the algorithm will fail with probability at most $4\exp(-3) \leq 1/3$. □

Lemma 4.1.9. *If $\ell > 0$ and the total weight of uncolored vertices inside B_i's is larger than $\varepsilon/4$, then the input is rejected by Step 2 of Algorithm 4.1.1 with probability at least* $2/3$.

Proof. For the analysis, we partition X into two sets, a set X_1 with $16k/\varepsilon$ vertices, and a set X_2 with $32k/\varepsilon$ vertices. Note that X_1 and X_2 are independently random.

The probability that X_1 does not contain any uncolored vertex inside some B_i is at most $(1 - \varepsilon/4)^{16k/\varepsilon} < \exp(-4)$. Suppose that X_1 contains an uncolored vertex w inside a B_i. There exist two disjoint sets $V_1, V_2 \subseteq V_i$, each of weight at least $\varepsilon/16k$, such that every path between two vertices $v_1 \in V_1$ and $v_2 \in V_2$ passes through w. It is enough to sample one vertex from each of these sets in order to reject the input in Step 2. The probability that at least one of these sets does not intersect X_2 is at most $2(1 - \varepsilon/16k)^{\frac{32k}{\varepsilon}} < 2\exp(-2)$. Thus, by the union bound, the algorithm will fail with probability at most $\exp(-4) + 2\exp(-2) < 1/3$. □

Lemma 4.1.10. *If $\ell > 0$, the B_i's are disjoint, F is bad, and the input is not rejected in Step 2 or 3 of Algorithm 4.1.1, then it is rejected in Step 5 with probability at least* $2/3$.

Proof. Since F is bad, by Lemma 4.1.6, the weight of bad vertices in F is at least $\varepsilon/8$. Thus, the probability of not querying a bad vertex w in F in Step 4 is at most $(1-\varepsilon/8)^{16/\varepsilon} < \exp(-2) < \frac{1}{7}$.

Suppose now that we have chosen a bad vertex w in Step 4. 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, $\mu_i(T_w) < \frac{\varepsilon}{16k}$, as otherwise w would have been i-balanced. We thus have $\mu_i(V \setminus T_w) > \frac{\varepsilon}{4k} - \frac{\varepsilon}{16k} > \frac{\varepsilon}{8k}$, and hence the probability that X does not contain an i-vertex outside T_w is at most $(1-\varepsilon/8k)^{48k/\varepsilon} < \exp(-6) < \frac{1}{400}$.

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 $(1-\varepsilon/8)^{\log_{1/(1-\varepsilon/8)} 8} = \frac{1}{8}$.

To conclude, we can expect a bad vertex w associated with an abundant color i to be chosen in Step 4, with an i-vertex queried outside T_w in Step 1 and an i-vertex queried inside T_w in Step 4. In such a case the algorithm will detect a forbidden subpath and reject the input in Step 5. By the union bound, the probability of failure in this is at most $\frac{1}{7} + \frac{1}{400} + \frac{1}{8} < \frac{1}{3}$. □

Proof of Theorem 4.1.2. Lemmas 4.1.7, 4.1.8, 4.1.9, 4.1.10, together with Observation 4.1.4, yield that every quasi k-coloring that is ε-far from being quasi-convex is rejected by the algorithm with probability at least $2/3$, which completes the proof. □

4.1.1 Implementation of the computational step in Algorithm 4.1.1

The procedure for detecting forbidden subpaths with respect to quasi-convexity is very similar to the one presented in Section 3.1.1 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 4.1.11. *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 3.1.8.

Lemma 4.1.12. *For every vertex v the following holds:*

1. *If Procedure 4.1.11 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 4.1.11 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 3.1.7, we may run Procedure 4.1.11 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 $\widetilde{O}(k/\varepsilon^2)$ (quasi-linear in the maximum sample size), if we use a preprocessing stage of time $O(n)$. See Section 3.1.1 for details.

4.1.2 Testing quasi-convexity under constraints

We now discuss the problem of testing for quasi-convexity with constraints, as we did for convexity in Section 3.1.2.

Given a graph $G = (V, E)$, a quasi k-coloring $c : V \to \{0, \ldots, k\}$, a set $D \subseteq V$ of "constraints" and a weight function $\mu : V \setminus D \to \mathbb{R}$, then we say that c is *ε-close to quasi-convexity under D* if there exists a quasi-convex coloring $c' : V \to \{0, \ldots, k\}$ which agrees with c on the values of the vertices in D and whose restriction to $V \setminus D$ is ε-close to that of c. If c is not ε-close to quasi-convexity under D, we say that c is *ε-far from quasi-convexity under D*.

Similarly to testing convexity under constraints, we show that for a domain tree $T = (V, E)$, it is enough to test for quasi-convexity under a set of constraints D by adding D to the query set of the quasi-convexity algorithm (Algorithm 4.1.1). We also need to increase our sample sizes by constant factors, since the presence of uncolored constraint vertices makes testing for farness a bit more delicate.

Algorithm 4.1.13. *Identical to Algorithm 4.1.1, except for the following:*

- *The size of the sample set X in Step 1 is $\lceil 96k/\varepsilon \rceil$.*
- *At the end of Step 1 we also query all the constraint vertices in D.*
- *In Step 4 we sample $\lceil 32/\varepsilon \rceil$ vertices w independently.*
- *For every uncolored vertex w queried in Step 4, we sample $\lceil \log_{1/(1-\varepsilon/16)8} \rceil$ vertices in each tree T_w^i.*

Proposition 4.1.14. *For every $\varepsilon > 0$, Algorithm 4.1.13 is a 1-sided ε-test for quasi-convexity of quasi k-colorings of trees under a set of constraints D. The query complexity of the test is $O\left(k/\varepsilon^2 + |D|\right)$ and the time complexity is $O(kn/\varepsilon)$. This can be implemented in time $\widetilde{O}\left(k/\varepsilon^2 + |D|\right)$ with a preprocessing stage of time $O(n^2)$.*

Clearly, the computational complexity corresponds to that of Algorithm 4.1.1 (see Section 4.1.1). If we use preprocessing, then the computational complexity now depends on the size of the query set plus the constraint set, whereas, without preprocessing, the complexity depends only on the number of vertices in the tree. Hence, the computational complexity demands are satisfied. Also, Algorithm 4.1.13 never rejects a quasi-convex coloring.

The proof that every coloring which is ε-far from quasi-convexity under D is rejected by the algorithm with probability at least 2/3 is similar to that of Theorem 4.1.2, while using ideas from the proof of Proposition 3.1.13. Define the sets B_i of balanced vertices as in the proof of Theorem 4.1.2. Note that the weight function is defined only on $V \setminus D$. Lemma 4.1.3 and Observation 4.1.4 are clearly also true here.

Now, as in the proof of Proposition 3.1.13, one can see that Algorithm 4.1.13 rejects with high probability if the sets B_i are not disjoint or if there exists an i-vertex $w \in D$ such that $w \in B_j$ for $i \neq j$. Also similar to that proof, the algorithm rejects with high probability if the set of B_i's and constraint vertices implies a forbidden subpath (now in its quasi-convexity sense). Otherwise, we augment every B_i with all the i-colored constraint vertices and the vertices on the paths from B_i to them. For constraint vertices colored with a non-abundant color i, we create a new set B_i 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 B_i for every color i, where the sets are disjoint and every B_i contains only i-colored constraint vertices.

Lemma 4.1.15. *If the total weight of uncolored vertices inside the* extended B_i*'s is larger than* $\varepsilon/4$*, then the input is rejected in Step* 2 *with probability at least* 2/3*.*

Proof. Similar to that of Lemma 4.1.9. Again we show that with high probability, an uncolored vertex w in B_i is sampled. If w is i-balanced, then the proof is identical to that of Lemma 4.1.9. Otherwise, w is on a path between two i-colored constraint vertices, or between an i-balanced vertex and an i-colored constraint vertex, which can only increase the probability of discovering a forbidden subpath, leading to rejection. □

Now, enumerate the B_i's as $B_1, \ldots, B_\ell$. Clearly, if $\ell = 0$ then the restriction of c to $V \setminus D$ is ε-close to the quasi-convex coloring in which all the vertices are uncolored. We now assume that $\ell > 0$. For every B_i, define M_i, as in the proof of Algorithm 4.1.13, to be the extension of B_i with "outsider" vertices. We have

Lemma 4.1.16.

$$\sum_{i \text{ is abundant}} \mu_i(V \setminus M_i) \leq \varepsilon/4.$$

Proof sketch. The proof is essentially the same as that of Lemma 4.1.5. Note that the proof relies only on the fact that for every abundant color i, M_i contains all

the i-balanced vertices and heavy i-vertices. Therefore, the proof applies also here. □

We now define the forest F, as in the proof of Theorem 4.1.2, to be the forest consisting of all outsider vertices. We define good uncolored vertices in F and the monotonicity of F as we did there. However, when considering whether F can be made monotone, we must take the uncolored constraint vertices possibly in F into account (recall that colored constraint vertices are all contained in the B_i's and therefore are not in F). Therefore, F is called *good* if it can be made monotone by changing the color of satellite and uncolored vertices of weight at most $\varepsilon/4$, and any amount of other vertices in F, excluding constraint vertices. Otherwise, we say that F is *bad*.

We say that an uncolored vertex w in F is *good* if $\mu_i(T_w) \leq \frac{\varepsilon}{16}\mu(T_w)$, 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 a constraint uncolored vertex in F as an ancestor.

Lemma 4.1.17. *If the total weight of bad uncolored vertices in F is at most $\frac{\varepsilon}{16}$ and the total weight of obstacle vertices is at most $\frac{\varepsilon}{8}$ 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 $\varepsilon/16$. 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 $\varepsilon/8$. 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 $\varepsilon/16$ of the weight of any good subtree, the total weight of satellite vertices thus changed is at most $3\varepsilon/16$. Hence, we have changed a total weight of at most $\varepsilon/4$ of uncolored and satellite vertices, and therefore F is good. □

Lemma 4.1.18. *If $\ell > 0$, the B_i's are disjoint, and F is bad, then the algorithm rejects with probability at least* $2/3$.

Proof. By Lemma 4.1.17, either the weight of obstacle vertices in F is larger than $\varepsilon/8$ or the weight of bad uncolored vertices in F is larger than $\varepsilon/16$.

If the weight of obstacle vertices in F is larger than $\varepsilon/8$, then clearly an obstacle vertex u is sampled in Step 1 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 1 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, and thus the algorithm will certainly reject the input in Step 2, after discovering the forbidden subpath $\langle u, w, v\rangle$. 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 sample set of Step 1 (excluding constraint vertices), X, into two sets, a set X_1 with $48k/\varepsilon$ vertices, and a set X_2 with $48k/\varepsilon$ 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 $(1-\varepsilon/8)^{48k/\varepsilon} < \exp(-6k) \leq \exp(-6)$. 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, $\mu_i(V \setminus T_w) > \varepsilon/8k$. Therefore, the probability that X_2 does not contain an i-vertex v outside T_w is at most $\exp(-6)$. Hence, with probability at least $2/3$, a forbidden subpath $\langle u, w, v\rangle$ is discovered and the algorithm rejects in Step 2.

If the weight of bad uncolored vertices in F is larger than $\varepsilon/16$, then the lemma is proved almost identically to Lemma 4.1.10. □

To complete the proof of Proposition 4.1.14, we now establish a lemma similar to Lemma 4.1.7.

Lemma 4.1.19. *If $\ell > 0$ and all of the following conditions apply:*

(a) *The B_i's are all disjoint;*

(b) *There is no uncolored constraint vertex inside a B_i ;*

(c) *The set of extended B_i's and constraint vertices does not imply a forbidden subpath;*

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

(e) *F is good;*

then c is ε-close to quasi-convexity under D.

Proof. The proof is very similar to that of Lemma 4.1.7.

We show that there exists a quasi-convex coloring c' of T which agrees with c on D and whose restriction to $V \setminus D$ is ε-close to that of c. Define c' as follows. For every extended B_i, color all the vertices of the extended B_i with i. Then choose a monotone coloring of F which changes a minimum weight of uncolored and satellite vertices without coloring 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, excluding satellites, whose color has been changed. From Lemma 4.1.16, the total weight of such vertices is at most $\varepsilon/4$. Now consider uncolored vertices inside extended B_i's. From Condition (d), the total weight of such vertices is at most $\varepsilon/4$. As for satellites and uncolored vertices in F, from Condition (e), the total weight of these is at most $\varepsilon/4$. Finally, for non-abundant colors which may have changed, since they are of weight smaller than $\varepsilon/4k$ each, their total weight is smaller than $\varepsilon/4$. We conclude that c' is ε-close to c. □

Proof of Proposition 4.1.14. As discussed in the beginning of the proof, Algorithm 4.1.13 rejects the input with probability at least 2/3 if any of the conditions (a)-(c) in Proposition 4.1.19 is not satisfied. By Lemmas 4.1.15 and 4.1.18, it does so also if conditions (d) or (e), respectively, are not satisfied. Therefore, Algorithm 4.1.13 rejects every ε-far input with probability at least 2/3. □

4.2 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 domain 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.

4.2.1 ℓ-convexity of trees

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

Theorem 4.2.1. *There exists a 1-sided test for ℓ-convexity of k-colorings of trees with query complexity $\widetilde{O}(\ell/\varepsilon)$ 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 loose its distribution-free quality.

We next introduce the subroutines used for testing interesting trees.

Observation 4.2.2. *Given a subtree T' of T, a color i and $0 < p < 1$, there exists an algorithm whose query and computational complexity are both $O(\log(1/p)\varepsilon^{-1})$, 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 $2\ln(1/p)\varepsilon^{-1}$ 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 4.2.3. *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 $O(\log(1/p)\varepsilon^{-1})$ 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 3.1.12 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 $8\ln 12\varepsilon^{-1}$ 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 $x = \mathrm{Junc}(u, v, w_2)$. 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 3.1.12 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 Proposition 3.1.13 $T_{(u,v)}$ 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 $8 \log_3(1/p) \ln 12\varepsilon^{-1}$ 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 3.1.1). 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 4.2.4.

- *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 4.2.5. *Consider an iteration of the while loop in Algorithm 4.2.4. 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 4.2.6. *The while loop of Algorithm 4.2.4 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 4.2.1. By the first part of Lemma 4.2.5, 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 4.2.5, 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 4.2.4 is a 1-sided test for ℓ-convexity.

By Observation 4.2.2 and Lemma 4.2.3, the test in Step 2 can be implemented with query complexity $O(\log(\ell)/\varepsilon)$. Therefore, the total query complexity of the algorithm is $O(\ell \log(\ell)/\varepsilon)$. The computational complexity follows from Observation 4.2.2 and Lemmas 4.2.3 and 4.2.6. □

4.2.2 ℓ-quasi-convexity of trees

In this subsection we prove the following theorem.

Theorem 4.2.7. *There exists a 1-sided test for ℓ-quasi-convexity of quasi k-colorings of trees whose query complexity is $\widetilde{O}(\ell/\varepsilon^2)$ 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 constraints or the test for homogeneity as a subroutine, we use the test for quasi-convexity with constraints (see Section 4.1.2) 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 4.2.8. *Given a dangling subtree T_u of T and $0 < p < 1$, there exists an algorithm whose query and computational complexity are both $O(\log(1/p)\varepsilon^{-2})$, 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 4.2.3, we run Algorithm 4.1.1 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 Proposition 4.1.14. Finding the required witnesses is done with techniques similar to those used in Lemma 4.2.3. □

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 4.2.9. *Given a pinned subtree $T_{(u,v)}$ of T and $0 < p < 1$, there exists an algorithm with query complexity $O(\log(1/p)\varepsilon^{-2})$ 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 Proposition 4.1.14. in a similar manner as Lemma 4.2.3 follows from Proposition 3.1.13. 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 4.2.10. *The algorithm is the same as Algorithm 4.2.4, 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 4.2.11. *Consider an iteration of the while loop in Algorithm 4.2.10. 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 4.2.5. □

Lemma 4.2.12. *The while loop of Algorithm 4.2.4 runs at most* 5ℓ *times.*

Proof. As in the case of Algorithm 4.2.4, 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 4.2.7. We show that Algorithm 4.2.10 satisfies the stated requirements.

By the first part of Lemma 4.2.11, 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 4.2.11, 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 4.2.10 is a 1-sided test for ℓ-quasi-convexity.

By Observation 4.2.8 and Lemma 4.2.9, the test in Step 2 can be implemented in query complexity $O(\log(\ell)/\varepsilon^2)$. Therefore, the total query complexity of the algorithm is $O(\ell \log(\ell)/\varepsilon^2)$. The computational complexity follows from Observation 4.2.8 and Lemmas 4.2.9 and 4.2.12. 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. □

4.2.3 List convexity and list quasi-convexity of trees

Theorem 4.2.13. *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 $\widetilde{O}(\ell/\varepsilon)$ and computational complexity $O(\ell n)$, where $\ell = \sum_{i=1,\ldots,k} l_i$.*

Theorem 4.2.14. *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 $\widetilde{O}(\ell/\varepsilon^2)$ and computational complexity $O(\ell n)$, where $\ell = \sum_{1 \le i \le k,\ l_i < \infty} l_i$.*

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 $l_i < \infty$.

Chapter 5

Testing for Forbidden Posets in Ordered Rooted Forests

In this chapter, our domain tree is a rooted ordered forest $F = (V, E)$ (see Section 2.5). We henceforth omit the word "rooted", as we only discuss rooted forests. Throughout the chapter, our input is a k-coloring $c : V \to [k]$ of F. Given a set S of forbidden posets (namely, every poset is an ordered forest with a k-coloring), we are interested in the property of not containing an induced subposet from S, denoted by $\mathcal{P}_S$.

A *subforest* of F is an induced subgraph of F which preserves all the ancestry and left-to-right relation pairs. For any $v \in V$, let T_v be the subtree of F rooted in v. Let F_v be the subforest obtained from T_v by removing v. We call F_v the *subforest of* v. We further define the *rightforest of* v, denoted by F_v^R, as the subforest of F comprising of all the vertices that are incomparable to v and are positioned right of v.

Given a subforest F' in F, let $c|_{F'}$ denote the restriction of the coloring c to F'. As our tests are recursive, we often consider properties of restrictions of c to subforests of F. Hence, for clarity, instead of saying that a coloring c' of a subforest F' satisfies (or close to, or far from) a property $\mathcal{P}_{S'}$, we say that (F', c') satisfies (or close to, or far from) $\mathcal{P}_{S'}$. Whenever c' is obvious, we may abuse notation and refer to F' instead of (F', c').

5.1 A test for a set of chains

In this section we consider a set S of forbidden chains with respect to the ancestry relation. In other words, (F, c) satisfies $\mathcal{P}_S$ if and only if it does not contain a chain in S as a subpath of a path from a root to a leaf. We note that monotonicity is a special case of such a property, and our algorithm is in some sense a generalization of the one presented in [18] for Boolean monotonicity in rooted trees.

Denote $S = \{s_1, \ldots, s_m\}$ and $\mathcal{S} = \Sigma_{j=1}^{m}|s_j|$. For every $j \in [m]$, let r_j denote the color of the first (top) node of s_j. Let $R \stackrel{\text{def}}{=} \bigcup_{j=1,\ldots,\mathcal{S}}\{r_j\}$. A vertex $v \in V$ is called an *i-vertex* if $c(v) = i$ and an *R-vertex* if $c(v) \in R$. For every $i \in [k]$, let $S^{(i)}$ be the set derived from S by removing the first node of all chains beginning in i. Formally, $S^{(i)} \stackrel{\text{def}}{=} \{s_j \mid r_j \neq i\} \cup \{x_j \mid s_j = ix_j\}$.

In the following recursive algorithm we assume that we can determine the highest root in F in time $O(1)$. In addition, for every vertex v in F, we assume that we can determine a highest root of a tree in F_v in time $O(1)$. This information can be stored in a preprocessing stage of traversing the forest in time $O(n)$.

We first present an adaptive algorithm, and later show how to transform it into a non-adaptive one, while maintaining a query complexity that is similar to that of the adaptive test.

Algorithm 5.1.1. Chain-Test(F, S, ε, c)

1. *If S includes an empty chain, reject.*
2. *Otherwise, if F or S is empty, accept.*
3. *Otherwise, if there exists $b \in [k]$ such that $b \notin R$: Query $\frac{32}{\varepsilon}$ vertices in F, where each vertex is independently chosen according to the distribution defined by μ. Reject if there exists a sampled vertex v which is an i-vertex v for some $i \in R$ and for which* Chain-Test(F_v, $S^{(i)}$, $\varepsilon/2$, $c|_{F_v}$) *has rejected. Otherwise, accept.*
4. *Otherwise, if $R = [k]$: Query $\frac{32}{\varepsilon}$ roots in F. The roots will be chosen independently, with possible repetitions, where the probability of selecting a certain root R is proportional to the total weight of vertices in T_r. In addition, query the root of a highest tree in the forest, i.e., a tree containing a longest path from the root to a leaf. Reject if the sample contains an i-vertex v for some $i \in [k]$, such that* Chain-Test(F_v, $S^{(i)}$, $\varepsilon/2$, $c|_{F_v}$) *has rejected, and accept otherwise.*

Theorem 5.1.2. Chain-Test *is a 1-sided ε-test for $\mathcal{P}_S$ for every $\varepsilon > 0$. The query complexity is*

$$\varepsilon^{-\mathcal{S}} \cdot 2^{\mathcal{S}^2/2+O(\mathcal{S})}$$

and is independent of the number of colors k. The time complexity of the algorithm is $O(n)$ for the preprocessing stage plus the time it takes to make the queries.

Proof. We begin by proving the complexity bounds. Let $Q(\mathcal{S}, \varepsilon)$ be a bound on the number of queries performed when the algorithm is called with parameters S and ε such that the sum of chain lengths in S is $\mathcal{S}$, and assume that Q is monotone in $\mathcal{S}$. It is easy to see that for $\mathcal{S} > 0$ we have $Q(\mathcal{S}, \varepsilon) \leq \left(\frac{32}{\varepsilon} + 1\right)\left(Q\left(\mathcal{S} - 1, \frac{\varepsilon}{2}\right) + 1\right)$, and since $\varepsilon \leq 1$ we have $Q(\mathcal{S}, \varepsilon) \leq \frac{33}{\varepsilon}\left(Q\left(\mathcal{S} - 1, \frac{\varepsilon}{2}\right) + 1\right)$. For the sake of clarity, we assume that $Q(0, \varepsilon) = 1$ for every $\varepsilon > 0$ (although in the algorithm above the actual number of queries for $\mathcal{S} = 0$ is zero), and therefore $Q(\mathcal{S}, \varepsilon) \geq 1$ for every $\mathcal{S} \geq 0, \varepsilon > 0$. Hence we obtain

$$\begin{aligned} Q(\mathcal{S}, \varepsilon) &\leq \frac{66}{\varepsilon} Q\left(\mathcal{S} - 1, \frac{\varepsilon}{2}\right) \leq \frac{66}{\varepsilon}\frac{66}{\varepsilon/2} Q\left(\mathcal{S} - 2, \frac{\varepsilon}{4}\right) \leq \frac{66}{\varepsilon}\frac{66}{\varepsilon/2}\frac{66}{\varepsilon/4} Q\left(\mathcal{S} - 3, \frac{\varepsilon}{8}\right) \\ &\leq \cdots \leq \left(\frac{66}{\varepsilon}\right)^j 2^{1+2+\ldots+(j-1)} \cdot Q\left(\mathcal{S} - j, \frac{\varepsilon}{2^j}\right). \end{aligned}$$

Substituting for $j = \mathcal{S}$ we have

$$Q(\mathcal{S}, \varepsilon) \leq \left(\frac{66}{\varepsilon}\right)^{\mathcal{S}} 2^{(\mathcal{S}-1)(\mathcal{S}-2)/2} = \varepsilon^{-\mathcal{S}} \cdot 2^{\mathcal{S}^2/2+O(\mathcal{S})}.$$

The dominant element in the time complexity of the algorithm is the preprocessing traversal of the tree in time $O(n)$. The time complexity of other operations is the same as the time required for making the queries (we neglect the time for calculating $S^{(i)}$ from S). Therefore, the complexity requirements are fulfilled.

We prove the correctness of the algorithm by induction on $\mathcal{S}$. The induction is trivial for Cases 1 and 2, so we will next prove it for Cases 3 and 4.

Lemma 5.1.3. *If the input coloring c satisfies $\mathcal{P}_S$ then* Chain-Test *accepts with probability 1.*

Proof. Suppose that the input was rejected. Whether the algorithm has executed Case 3 or 4, an i-vertex v was found for some $i \in [k]$ such that Chain-Test(F_v, $S^{(i)}$, $\varepsilon/2$, $c|_{F_v}$) has rejected. By the induction hypothesis, F_v does not satisfy $\mathcal{P}_{S^{(i)}}$ and

thus contains a chain $x \in S^{(i)}$. It clearly follows that F contains a chain $ix \in S$, and hence does not satisfy $\mathcal{P}_S$. $\square$

An i-vertex v is called *bad* if F_v is $\frac{\varepsilon}{2}$-far from $\mathcal{P}_{S^{(i)}}$. Otherwise it is called *good*. The next two lemmas deal with Case 3 of the algorithm.

Lemma 5.1.4. *If the total weight of bad R-vertices is less than $\varepsilon/2$ and Case 3 of the algorithm applies, then (F, c) is ε-close to $\mathcal{P}_S$.*

Proof. Let G be the set of good R-vertices in F. Let M_G be the set of vertices in G which have no proper ancestors in G.

We now show that there exists a coloring c' of F that is ε-close to c such that (F, c') satisfies $\mathcal{P}_S$. By definition, for every i-vertex $v \in M_G$, there exists a coloring c_v of F_v which is $\varepsilon/2$-close to $c|_{F_v}$, such that (F_v, c_v) satisfies $\mathcal{P}_{S^{(i)}}$. Since no vertex in M_G is an ancestor of another, the F_v's are all disjoint. We thus set $c'(u) = c_v(u)$ for any proper descendant u of $v \in M_G$. Clearly, this changes the color vertices within subforests of good R-vertices, whose total weight is at most $\varepsilon/2$.

For every bad R-vertex v which is not a descendant of a vertex in M_G, we let $c'(v) = b$, where $b \notin R$. The rest of the vertices remain with their original coloring. Since the total weight of bad R-vertices is at most $\varepsilon/2$, c' is ε-close to c. Now, the only R-vertices in the recolored forest are in subtrees of vertices in M_G. However, for every i-vertex $v \in M_G$, F_v does not contain any chain in $S^{(i)}$. Therefore, c' satisfies $\mathcal{P}_S$. $\square$

Lemma 5.1.5. *If the total weight of bad R-vertices is at least $\varepsilon/2$ and Case 3 of the algorithm applies, then (F, c) is rejected with probability at least $\frac{2}{3}$.*

Proof. Using the Chernoff Bound, the probability of sampling less than 8 bad R-vertices is smaller than $e^{-2} < \frac{1}{6}$. Assume that we sampled at least 8 bad R-vertices. By the induction hypothesis, for every bad i-vertex v, Chain-Test(F_v, $S^{(i)}$, $\varepsilon/2$,$c|_{F_v}$) accepts with probability at most $\frac{1}{3}$. Since the tests are independent, the probability that all the tests accept is at most $\left(\frac{1}{3}\right)^8 < \frac{1}{6}$. We conclude that the input is accepted with probability at most $\frac{1}{3}$. $\square$

The rest of the proof deals with Case 4 of the algorithm. For a subforest F' or F and a set S' of forbidden chains, we say that F' is *S'-avoidable* if there exists a coloring c' of F' such that $(F', c'|_{F'})$ satisfies $\mathcal{P}_{S'}$. Otherwise, we say that F' is *S'-unavoidable*.

Observation 5.1.6. *If F is S-avoidable then every subtree in F is S-avoidable.* □

Lemma 5.1.7. *Suppose that F is S-avoidable and, in addition, the total weight of vertices in trees whose roots are bad is less than $\varepsilon/2$. Then, if Case 4 of the algorithm applies, then (F, c) is ε-close to $\mathcal{P}_S$.*

Proof. We show that there exists a coloring c' of F that satisfies $\mathcal{P}_S$ and is ε-close to c. Let M_G be the set of good roots in F. For every $v \in M_G$, by definition, there exists a coloring c_v of F_v which is $\varepsilon/2$-close to the restriction of c to F_v, such that (F_v, c_v) satisfies $\mathcal{P}_{S^{(i)}}$. Since no vertex in M_G is an ancestor of another, the F_v's are disjoint. We thus set $c'(u) = c_v(u)$ for any proper descendant u of $v \in M_G$. Clearly, this changes the color of vertices within subforests of good vertices, and their total weight is at most $\varepsilon/2$.

The total weight of vertices in trees whose roots are bad is at most $\varepsilon/2$. As F is S-avoidable, by Observation 5.1.6, any of these trees is also S-avoidable. We therefore color every tree T whose root is bad according to a coloring c_T such that (T, c_T) satisfies $\mathcal{P}_S$. Since we have recolored additional vertices whose total weight is at most $\varepsilon/2$, the resulting coloring c' is ε-close to c. As we have taken care of trees whose roots are good as well as of trees whose roots are bad, it follows that c' satisfies $\mathcal{P}_S$. □

Observation 5.1.8. *If a highest tree in F is S-avoidable then F is S-avoidable.*

Proof. Suppose that a highest tree T in F is S-avoidable. Therefore, there exists a coloring c_T of T such that (T, c_T) satisfies $\mathcal{P}_S$. We define a coloring c of F as follows. Let ℓ be a longest path from the root of T to a leaf. Let the *level* of a vertex be its distance to a root. We color every vertex of level j in F with the same color as the vertex of level j in ℓ is colored under c_T. As ℓ is a longest path, c is well defined, and, in addition, the restriction of c to ℓ coincides with c_T. Now suppose that (F, c) contains a chain in S. Then this chain is contained within a path from a root to a leaf. However, by the definition of c, a chain with the same colors is contained in (ℓ, c_T), in contradiction to the assumption that (ℓ, c_T) satisfies $\mathcal{P}_S$. □

Lemma 5.1.9. *Suppose that Case 4 of the algorithm applies. If F is S-unavoidable, or the total weight of vertices in trees whose roots are bad is at least $\varepsilon/2$, then (F, c) is rejected with probability at least $\frac{2}{3}$.*

Proof. If F is S-unavoidable, then, by Observation 5.1.8, any highest tree in F is S-unavoidable. In particular, the highest tree whose root is queried by Case 4 of

Algorithm 5.1.1 is 1-far from $\mathcal{P}_S$. Let v be the root of that tree and let i be the color of v. By the induction hypothesis, Chain-Test(F_v, $S^{(i)}$, $\varepsilon/2$, $c|_{F_v}$) rejects with probability at least $\frac{2}{3}$.

For the case where the total weight of vertices in trees whose roots are bad is at least $\varepsilon/2$, the proof is essentially the same as for Lemma 5.1.5. Since the probability of choosing a certain root is proportional to the total weight of its tree, the probability of choosing a bad root in each iteration in this case is at least $\varepsilon/2$, as was the probability of choosing a bad vertex in Case 3 of the algorithm, on which the Chernoff bound was applied. □

This completes the proof of Theorem 5.1.2. □

5.1.1 A non-adaptive chain test

In order to transform Algorithm 5.1.1 into a non-adaptive one, note that the distribution of vertices sampled in a specific recursive call of the test depends only on which of the four cases of the algorithm was executed. Therefore, one may decide what vertices to query regardless of previous answers, for any possible scenario of the cases of Algorithm 5.1.1. We thus define a non-adaptive procedure where we perform our queries. Since we do not know what set S of chains will be actually used for the queries, we use an upper bound on the sum of chain lengths in S, denoted by U, instead of the actual set S.

Algorithm 5.1.10. Query(F, U, ε)
We define the algorithm by induction on U.

- *If $U = 0$, do nothing.*
- *Otherwise:*
  1. *Query $\frac{32}{\varepsilon}$ vertices in F, where each vertex is independently chosen according to the distribution defined by μ. For each queried vertex v, call* Query(F_v, $U - 1$, $\varepsilon/2$).
  2. *Query $\frac{32}{\varepsilon}$ roots in F. The roots will be chosen independently, with possible repetitions, where the probability of selecting a certain root is proportional to the total weight of the tree rooted in it. In addition, query the root of a highest tree in F. For each queried vertex v, call* Query(F_v, $U - 1$, $\varepsilon/2$).

After calling Query(F, U, ε) with $U = \mathcal{S}$ and the original ε, we may simulate the running of Algorithm 5.1.1 by simply using the relevant answers from the queries produced in every recursion level. Clearly, we will have all our required answers, as the depth of the recursion is at most $\mathcal{S}$. By applying similar techniques to those used for the adaptive case, we obtain that the query and time complexity for the non-adaptive test are essentially the same, that is, $\varepsilon^{-\mathcal{S}} \cdot 2^{\mathcal{S}^2/2+O(\mathcal{S})}$ queries and $O(n)$ time, where the only difference is in the coefficient hidden in the O notation.

5.2 A test for a forbidden forest

In this section we consider the property $\mathcal{P}_S$ where $S = \{f\}$ for some colored ordered forest f. By a slight abuse of notation we refer to this property as $\mathcal{P}_f$ rather than $\mathcal{P}_{\{f\}}$. Given a colored vertex r in f, we use the notations f_r and f_r^R to denote the subforest of r and the right subforest of r with respect to f, analogously as we did for F. However, here we refer to a colored ordered forest.

For clarity, we first present an adaptive algorithm, and later show how to transform it into a non-adaptive algorithm. Note that our algorithm uses no information on the number of colors k. On the other hand, denoting the size of f by $\mathcal{F}$, we may clearly assume that k is not larger than $\mathcal{F} + 1$.

Algorithm 5.2.1. Forest-Test(F, f, ε, c)

1. *If f is empty, reject.*
2. *Otherwise, if F is empty, accept.*
3. *Otherwise, let r be the leftmost root in f. Let i be the color of r. Query $\frac{32}{\varepsilon}$ vertices in F, where each vertex is independently chosen according to the distribution defined by μ. Reject if there exists a sampled i-vertex v such that both* Forest-Test(F_v, f_r, $\varepsilon/2$, $c|_{F_v}$) *and* Forest-Test(F_v^R, f_r^R, $\varepsilon/2$, $c|_{F_v^R}$) *have rejected, and accept otherwise.*

Theorem 5.2.2. Forest-Test *is a 1-sided ε-test for $\mathcal{P}_f$ for every $\varepsilon > 0$. The query complexity of the test is*

$$\varepsilon^{-\mathcal{F}} \cdot 2^{\mathcal{F}^2/2+O(\mathcal{F})}$$

and the time complexity is the time required for performing the queries.

Proof. We start with the complexity requirements. Let $Q(\mathcal{F}, \varepsilon)$ denote the query complexity when the algorithm is being called with parameters f and ε such that the size of f is $\mathcal{F}$. It can be seen that for $\mathcal{F} > 0$ we have

$$Q(\mathcal{F}, \varepsilon) \leq \frac{64}{\varepsilon}\left(Q\left(\mathcal{F}-1, \frac{\varepsilon}{2}\right)+1\right).$$

With the assumption that $Q(0, \varepsilon) = 1$ for every $\varepsilon > 0$ we have

$$Q(\mathcal{F}, \varepsilon) \leq \frac{128}{\varepsilon} Q\left(\mathcal{F}-1, \frac{\varepsilon}{2}\right) \leq \cdots \leq O\left(\left(\frac{128}{\varepsilon}\right)^{\mathcal{F}} 2^{\mathcal{F}^2/2}\right) = \varepsilon^{-\mathcal{F}} \cdot 2^{\mathcal{F}^2/2+O(\mathcal{F})},$$

using similar techniques to those used in Section 5.1, one can see that the time complexity is no larger than the time needed for making the queries.

The algorithm is trivially correct for the base cases. We henceforth prove the correctness of Case 3 by induction on $\mathcal{F}$.

Lemma 5.2.3. *If (F, c) satisfies $\mathcal{P}_f$ then* Forest-Test *accepts with probability 1.*

Proof. Suppose that the input was rejected. Then an i-vertex v exists such that both Forest-Test(F_v, f_r, $\varepsilon/2$, $c|_{F_v}$) and Forest-Test(F_v^R, f_r^R, $\varepsilon/2$, $c|_{F_v}^R$) have rejected. By the induction hypothesis, F_v contains f_r and F_v^R contains f_r^R. It clearly follows that F contains f, and hence does not satisfy $\mathcal{P}_f$. □

Let r be the leftmost root of f and let i denote its color. An i-vertex v in F is called *down-bad* (with respect to f) if F_v is $\varepsilon/2$-far from $\mathcal{P}_{f_r}$. Otherwise it is called *down-good*. v is called *right-bad* if F_v^R is $\varepsilon/2$-far from $\mathcal{P}_{f_r^R}$. Otherwise it is called *right-good*. We say that a vertex is *bad* if it is both down-bad and right-bad.

Lemma 5.2.4. *If the total weight of bad vertices is no larger than $\varepsilon/2$, then (F, c) is ε-close to $\mathcal{P}_f$.*

Proof. We show that in such a case there exists a coloring c' of F that satisfies $\mathcal{P}_f$ and is ε-close to c.

Let U be the set of down-bad i-vertices in F which are right-good. If U is not empty (one of the cases where U is empty is when f is a tree), let u be the first vertex in U according to the "post-order", that is, there are no vertices of U left of u or below it. By definition, there exists a coloring c_u of F_u^R which is $\varepsilon/2$-close to $c|_{F_u^R}$, such that (F_u^R, c_u) satisfies $\mathcal{P}_{f_u^R}$. We thus set $c'(w) = c_u(w)$ for every $w \in F_u^R$.

Note that there is a total weight of at most $\varepsilon/2$ of down-bad i-vertices left of u or below it (or in the entire forest, if u does not exist).

Let G be the set of down-good i-vertices to the left of u or below it (or all down-good i-vertices if u does not exist). Let M_G be the set of vertices in G which have no proper ancestors in G. By definition, for every $v \in M_G$ there exists a coloring c_v of F_v which is $\varepsilon/2$-close to $c|_{F_v}$, such that (F_v, c_v) satisfies $\mathcal{P}_{f_r}$. We thus set $c'(w) = c_v(w)$ for every proper descendant u of $v \in M_G$. Clearly, we have so far recolored vertices whose total weight is at most $\varepsilon/2$. For every down-bad i-vertex v which is to the left of u and is not a descendant of a vertex in M_G, we let $c'(v)$ be any color other than i. The rest of the vertices remain unchanged. Since the total weight of such vertices is at most $\varepsilon/2$ (as they are also right-bad), c' is ε-close to c.

To see that c' satisfies $\mathcal{P}_f$, note that the only i-vertices to the left of u or below it are descendants of vertices in M_G. However, the subforests of vertices in M_G do not contain f_r. On the other hand, f_r^R cannot be found to the right of u. It clearly follows that F does not contain f. □

Lemma 5.2.5. *If the total weight of bad vertices is at most $\varepsilon/2$, then (F, c) is accepted with probability at most $\frac{1}{3}$.*

Proof. Using the Chernoff Bound, the probability of sampling less than 8 bad vertices is smaller than $e^{-2} < \frac{1}{6}$. Assume that we sampled at least 8 bad vertices. By the induction hypothesis, for every bad vertex v, Forest-Test(F_v, f_r, $\varepsilon/2$, $c|_{F_v}$) and Forest-Test(F_v^R, f_r^R, $\varepsilon/2$, $c|_{F_v^R}$) both accept with probability at most $\frac{1}{3}$, and so at least one of these two tests accepts with probability at most $\frac{2}{3}$. Since the tests are independent, the probability that for every bad vertex v at least one test accepts is at most $\left(\frac{2}{3}\right)^8 < \frac{1}{6}$. We conclude that the input is accepted with probability at most $\frac{1}{3}$. □

This completes the proof of Theorem 5.2.2. □

5.2.1 A non-adaptive forest test

As in the test for chains, the distribution of vertices sampled in Case 3 is independent of the answers for previous queries. Therefore, we may choose all the queried vertices of the algorithm in advance. The essential difference between the adaptive and non-adaptive versions, is that in the adaptive version we perform the recursion on F_v only for queried vertices v which are i-vertices. In the non-adaptive case, we have

to perform the recursion for all queried vertices. Therefore, the query complexity for the non-adaptive test is the same as the worst case of the adaptive one, that is, $\varepsilon^{-\mathcal{F}} \cdot 2^{\mathcal{F}^2/2+O(\mathcal{F})}$. The time complexity is again the time it takes to make the queries.

5.3 Discussion

A test for a general set of forbidden forests would require more than separately testing for each of the forbidden forests, as a priori it may be the case that the input is close to not containing each of the forbidden forests separately, and yet far from simultaneously not containing any of them. Also, the test would have to be more sophisticated than our tests given in this chapter, in the sense that it is not enough to detect a far input by finding an instance of a forbidden forest. Indeed, it may be the case that a single vertex is common to all instances of forbidden forests, but any attempt to remove these instances by recoloring the vertex results in appearance of other instances of (possibly different) forbidden forests. We see an example of this situation in the cases of convexity and quasi-convexity, where our tests attempt to discover the existence of such a critical vertex without actually querying it. Another example is in our chain test, where for some inputs we have to query a longest path between a root to a leaf, to make sure that F has a coloring which contains no forbidden chain.

Another natural direction would be to consider rooted forests which are not ordered. Testing for a forbidden forest in an unordered rooted forest may clearly be reduced to testing for a set of forbidden rooted ordered forests. However, as the size of the forbidden set, in terms of the total number of vertices, may be up to a factorial in the size of the original forest, the question arises as to whether there is a more efficient way of solving the problem.

Part II

Testing Graph Orientations for Being Eulerian

Chapter 6

Introduction

6.1 Testing in the orientation model

Testing properties of directed graphs has been studied in the past through naturally adapting the dense graph model and the bounded degree model (see [4, 7]). Here we consider property testing of directed graphs in the *orientation model*, whose study began in [30] and continued in [31] and [12]. The orientation model is a massively parameterized model (see Section 1.3). The domain graph in this model is an undirected graph $G = (V, E)$, and the input is an orientation $\overrightarrow{G}$ of G, in which every edge in E has a direction. Our testers may access the input using edge queries. That is, every query concerns an edge $e \in E$, and the answer to the query is the direction of e in $\overrightarrow{G}$. Our distance function is the Hamming distance on G's orientations. Namely, an orientation $\overrightarrow{G}$ of G is called *ε-close* to a property $\mathcal{P}$ if it can be made to satisfy $\mathcal{P}$ by inverting at most an ε-fraction of the edges of G, and otherwise $\overrightarrow{G}$ is said to be *ε-far* from $\mathcal{P}$.

The orientation model provides a novel and interesting setting for research. Studying this model not only requires new techniques, but it also provides a natural and rich framework for exploring properties of directed graphs. In addition, testers for the orientation model could have practical counterparts in some physical networks, such as a road network, where it is plausible to assume that one can reverse the direction of an existing arc more cheaply than one can add or delete a new arc.

6.2 Eulerian graphs and Eulerian orientations

We consider the property of being Eulerian, which was presented in [31] as one of the natural orientation properties whose query complexity was still unknown. A directed graph $\overrightarrow{G}$ is called *Eulerian* if for every vertex v in the graph, the in-degree of v is equal to its out-degree (in addition, it is common to define Eulerian graphs as connected, but as we explain later, our algorithms and proofs work equally well whether we require connectivity or not). An undirected graph G has an Eulerian orientation $\overrightarrow{G}$ if and only if all the degrees of G are even. Such an undirected domain graph is called Eulerian also. Throughout our work we assume that our underlying undirected graph G is Eulerian.

Eulerian graphs and Eulerian orientations have attracted researchers since the dawn of graph theory in 1736, when Leonard Euler published his solution for the famous "Königsberg bridge problem". Throughout the years, Eulerian graphs have been the subject of extensive research (e.g. [45, 36, 52, 38, 11, 6]; see [22, 23] for an extensive survey). Aside from their appealing theoretic characteristics, Eulerian graphs have been studied in the context of networking [33] and genetics [44].

Testing for being Eulerian in the orientation model is equivalent to the following problem. We have a known network (a communication network, a transportation system or a piping system) where every edge can transport a unit of "flow" in both directions. Our goal is to know whether the network is "balanced", or far from being balanced, where being balanced means that the number of flows entering every node in the network is equal to the number of flows exiting it (so there is no "accumulation" in the nodes). To examine the network, we detect the flow direction in selected individual edges, which is deemed to be the expensive operation.

The main difficulty in testing orientations for being Eulerian arises from the fact that an orientation might have a small number of unbalanced vertices, and each of them with a small imbalance, and yet be far from being Eulerian. This is since trying to balance an unbalanced vertex by inverting some of its incident edges may violate the balance of its balanced neighbors. Thus, we must continue to invert edges along a directed path between a vertex with a positive imbalance and a vertex with a negative imbalance. We call such a path a *correction path*. A main component of our work is giving upper bounds for the length of the correction paths. In this context we note the work of Babai [6], who studied the ratio between the diameter of Eulerian digraphs and the diameter of their underlying undirected graphs. While he

gave an upper bound for this ratio for vertex-transitive graphs, he showed an infinite family of undirected graphs with diameter 2 which have an Eulerian orientation with diameter $\Omega(n^{1/3})$.

6.3 Our results

Our upper bounds are based on three "generic" tests, one 1-sided test and two 2-sided tests. Instead of receiving ε as a parameter, the generic tests receive a parameter p, which stands for the number of required correction paths in an orientation that is far from being Eulerian. We hence call these tests *p-tests*. We later derive ε-tests from the p-tests by proving two lower bounds for p. The first one gives an efficient test for dense graphs and the second one gives an efficient test for expander graphs. Finally, we show how to use variations of the expander tests for obtaining a 1-sided test and a 2-sided test for general graphs, using a decomposition ("chopping") procedure into subgraphs that are roughly expanders. The 2-sided test that we obtain this way has a sub-linear query complexity for every graph. Unfortunately, our chopping procedure is adaptive and has an exponential computational time in $|E|$. All of our other algorithms are non-adaptive and their computational complexity is of the same order as their query complexity.

On the negative side, we provide several lower bounds. We show that any 1-sided test for being Eulerian must use $\Omega(m)$ queries for some graphs. For bounded-degree graphs, we use the toroidal grid to prove non-constant 1-sided and 2-sided lower bounds. These bounds are noteworthy, as bounded-degree graphs have a constant size witness for not being Eulerian, namely the set of edges incident with one unbalanced vertex. In contrast, the *st*-connectivity property, whose witness must include a cut in the graph, is testable with a constant number of queries in the orientation model [12]. Other testing models also have known super-constant lower bounds for some properties which have constant-size witness. For instance, in [9] it is proved that testing whether a truth assignment satisfies a known 3CNF formula requires a linear number of queries for some formulas. However, most of these bounds are for properties that have stronger expressive power than that of being Eulerian.

The following chart gives a summary of our upper and lower bounds for the query complexity of ε-tests. Here and throughout Part II, we set $n = |V|$ and $m = |E|$, let Δ be the maximum vertex-degree in G, and set $d \stackrel{\text{def}}{=} m/n$.

Result	1-sided tests	2-sided tests
Tests for large d (Section 8.2)	$O\left(\frac{\Delta m}{\varepsilon^2 d^2}\right)$	$\min\left\{\widetilde{O}\left(\frac{m^3}{\varepsilon^6 d^6}\right), \widetilde{O}\left(\frac{\sqrt{\Delta} m}{\varepsilon^2 d^2}\right)\right\}$
Tests for α-expanders (Section 8.3)	$O\left(\frac{\Delta \log(1/\varepsilon)}{\alpha\varepsilon}\right)$	$\min\left\{\widetilde{O}\left(\left(\frac{\log(1/\varepsilon)}{\alpha\varepsilon}\right)^3\right), \widetilde{O}\left(\frac{\sqrt{\Delta}\log(1/\varepsilon)}{\alpha\varepsilon}\right)\right\}$
General tests (Section 8.5)	$O\left(\frac{(\Delta m \log m)^{2/3}}{\varepsilon^{4/3}}\right)$	$\min\left\{\widetilde{O}\left(\frac{\Delta^{1/3} m^{2/3}}{\varepsilon^{4/3}}\right), \widetilde{O}\left(\frac{\Delta^{3/16} m^{3/4}}{\varepsilon^{5/4}}\right)\right\}$
Simple lower bound (Section 7.4)	$\Omega(m)$	—
Lower bounds for bounded-degree graphs (Chapter 9)	$\Omega(\log m)$, $\Omega(m^{1/4})$ non-adaptive	$\Omega(\log\log m)$, $\Omega\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ non-adaptive

The rest of Part II is organized as follows. Chapter 7 provides general definitions and lemmas to be used in the sequel, including a simple 1-sided lower bound for general graphs (Section 7.4). Chapter 8 is dedicated to our upper bounds: Section 8.1 contains our three p-tests, which distinguish between Eulerian orientations and orientations with many correction paths. In Section 8.2 we give a lower bound on the number of correction paths as a function of the average degree in the graph, and derive our tests for graphs with high average degree. In Section 8.3 we give such a bound and derive tests for expander graphs. Section 8.4 considers testing subgraphs that we call "lame" expanders, providing results for them that are similar to those obtained for expanders, and are used in the sequel. Section 8.5 presents our most general tests, which use the results of Section 8.4. Section 8.6 concludes the section with a short discussion of the upper bounds results and open problems. To end this part of the thesis, Chapter 9 gives our 2-sided and 1-sided lower bounds for bounded-degree graphs.

Chapter 7

Basic Definitions and Lemmas

7.1 Preliminaries

Throughout Part II, we assume that our domain graph $G = (V, E)$ is an Eulerian undirected graph, that is, for every $v \in V$, the degree $\deg(v)$ of v is even.

Given an orientation $\overrightarrow{G} = (V, \overrightarrow{E})$ and a vertex $v \in V$, let $\text{indeg}_{\overrightarrow{G}}(v)$ denote the in-degree of v with respect to $\overrightarrow{G}$ and let $\text{outdeg}_{\overrightarrow{G}}(v)$ denote the out-degree of v with respect to $\overrightarrow{G}$. We define the *imbalance* of v in $\overrightarrow{G}$ as $\text{ib}_{\overrightarrow{G}}(v) \stackrel{\text{def}}{=} \text{outdeg}_{\overrightarrow{G}}(v) - \text{indeg}_{\overrightarrow{G}}(v)$. We sometimes omit the subscript $\overrightarrow{G}$ when it is obvious from the context. We say that a vertex $v \in V$ is a *spring* in $\overrightarrow{G}$ if $\text{ib}_{\overrightarrow{G}}(v) > 0$. We say that v is a *drain* in $\overrightarrow{G}$ if $\text{ib}_{\overrightarrow{G}}(v) < 0$. If $\text{ib}_{\overrightarrow{G}}(v) = 0$ then we say that v is *balanced* in $\overrightarrow{G}$. We say that $\overrightarrow{G}$ is *Eulerian* if all its vertices are balanced. Since all the vertices of G are of even degree, there always exists some Eulerian orientation $\overrightarrow{G}$ of G.

We note that it is common to require connectivity from an Eulerian undirected graph and strong connectivity from a directed Eulerian graph. However, all our algorithms and proofs work also for the case where our domain graph G is not connected. Furthermore, it is easy to see that if G is connected, then every Eulerian orientation $\overrightarrow{G}$ of G is strongly connected. Thus, if we are interested in testing whether $\overrightarrow{G}$ is strongly connected, in addition to having balanced vertices, we simply add to our tests a phase which checks the domain graph G and rejects if it is not connected. We henceforth ignore the connectivity criterion.

Note that testing whether $\overrightarrow{G}$ is Eulerian is trivial for $\varepsilon \geq \frac{1}{2}$.

Observation 7.1.1. *Every orientation $\overrightarrow{G}$ of G is $\frac{1}{2}$-close to being Eulerian.*

Proof. Let $\overrightarrow{G_1}$ be an arbitrary Eulerian orientation of G. Let $\overrightarrow{G_2} = \overleftarrow{G_1}$, namely,

the orientation derived from $\overrightarrow{G}_1$ by inverting all the edges. Clearly, $\overrightarrow{G_2}$ is Eulerian as well, since inverting all the edges maintains the absolute value of the imbalance of all vertices. Now, for every edge $e \in E$, the direction of e in $\overrightarrow{G}$ is the same as in $\overrightarrow{G_1}$ if and only if it is opposite to the direction of e in $\overrightarrow{G_2}$. Hence, $\mathrm{dist}(\overrightarrow{G}, \overrightarrow{G_1}) = 1 - \mathrm{dist}(\overrightarrow{G}, \overrightarrow{G_2})$, and therefore, $\overrightarrow{G}$ is $\frac{1}{2}$-close to either $\overrightarrow{G_1}$ or $\overrightarrow{G_2}$. $\square$

We conclude this section with some notation that is useful in the following. Given a set $U \subseteq V$, we let

$$E(U) \stackrel{\text{def}}{=} \{\{u,v\} \in E \mid u,v \in U\},$$

$$\overrightarrow{E}(U) \stackrel{\text{def}}{=} \{(u,v) \in \overrightarrow{E} \mid u,v \in U\},$$

$$\partial U \stackrel{\text{def}}{=} \{\{u,v\} \in E \mid u \in U, v \notin U\},$$

and

$$\overrightarrow{\partial} U \stackrel{\text{def}}{=} \{(u,v) \in \overrightarrow{E} \mid u \in U, v \notin U\}.$$

$$E(U,W) \stackrel{\text{def}}{=} \{\{u,w\} \in E \mid u \in U, w \in W\}$$

and

$$\overrightarrow{E}(U,W) \stackrel{\text{def}}{=} \{(u,w) \in \overrightarrow{E} \mid u \in U, w \in W\}.$$

7.2 Correction subgraphs and p-tests

Let $\overrightarrow{G}$ be an orientation of G. Given a subgraph $\overrightarrow{H} = (V_H, \overrightarrow{E_H})$ of $\overrightarrow{G}$ (that is, a directed graph where $V_H \subseteq V$ and $\overrightarrow{E_H} \subseteq \overrightarrow{E}$) we define $\overrightarrow{G}_{\overleftarrow{H}} \stackrel{\text{def}}{=} (V, \overrightarrow{E}_{\overleftarrow{H}})$ to be the orientation of G derived from $\overrightarrow{G}$ by inverting all the edges of $\overrightarrow{H}$. Namely,

$$\overrightarrow{E}_{\overleftarrow{H}} \;=\; \overrightarrow{E} \setminus \overrightarrow{E_H} \;\cup\; \{(v,u) \in (V_H)^2 \mid (u,v) \in \overrightarrow{E_H}\}.$$

We say that $\overrightarrow{H}$ is a *correction subgraph* of $\overrightarrow{G}$ if $\overrightarrow{G}_{\overleftarrow{H}}$ is Eulerian. Note that in such a case, $\overrightarrow{G}$ is $\frac{|\overrightarrow{E_H}|}{m}$-close to being Eulerian.

Lemma 7.2.1. *A subgraph $\overrightarrow{H}$ of $\overrightarrow{G}$ is a correction subgraph if and only if the following conditions hold for every $v \in V$:*

1. *If $v \notin V_H$, then v is balanced in $\overrightarrow{G}$.*
2. *If $v \in V_H$, then $\mathrm{ib}_{\overrightarrow{H}}(v) = \frac{1}{2} \cdot \mathrm{ib}_{\overrightarrow{G}}(v)$.*

In particular, a vertex v is a spring in $\overrightarrow{G}$ if and only if it is a spring in $\overrightarrow{H}$, and v is drain in $\overrightarrow{G}$ if and only if it is a drain in $\overrightarrow{H}$.

Proof. Remember that $\overrightarrow{H}$ is a correction subgraph of $\overrightarrow{G}$ if and only if $\mathrm{ib}_{\overrightarrow{G}_{\overleftarrow{H}}}(v) = 0$ for every $v \in V$. The proof follows from the following facts, which can be easily verified:

1. If $v \notin V_H$, then $\mathrm{ib}_{\overrightarrow{G}_{\overleftarrow{H}}}(v) = \mathrm{ib}_{\overrightarrow{G}}(v)$.
2. If $v \in V_H$, then $\mathrm{ib}_{\overrightarrow{G}_{\overleftarrow{H}}}(v) = \mathrm{ib}_{\overrightarrow{G}}(v) - 2 \cdot \mathrm{ib}_{\overrightarrow{H}}(v)$.

□

Since G is Eulerian, there exists some correction subgraph of $\overrightarrow{G}$. Furthermore, without loss of generality, we may focus on acyclic correction graphs.

Observation 7.2.2. *For any orientation $\overrightarrow{G}$ of G and for any correction subgraph $\overrightarrow{H_1}$ of $\overrightarrow{G}$, there exists an acyclic correction subgraph $\overrightarrow{H}$ of $\overrightarrow{G}$ that is a subgraph of $\overrightarrow{H_1}$.*

Proof. We obtain $\overrightarrow{H}$ from $\overrightarrow{H_1}$ as follows. While $\overrightarrow{H_1}$ is not acyclic, we arbitrarily choose a directed cycle and remove its edges from the subgraph. Note that this operation maintains the balance factors of all the vertices of G with respect to $\overrightarrow{H_1}$. The proof thus follows from Lemma 7.2.1. □

Let S be the set of springs in $\overrightarrow{G}$ and let T be the set of drains in $\overrightarrow{G}$. We say that a directed path $\overrightarrow{P} = \langle u_0, \ldots, u_k \rangle$ in $\overrightarrow{G}$ is a *spring-drain path* if $u_0 \in S$ and $u_k \in T$. Note that in this case, for any correction subgraph $\overrightarrow{H}$ of $\overrightarrow{G}$, we have by Lemma 7.2.1 that u_0 is a spring in $\overrightarrow{H}$ and u_k is a drain in $\overrightarrow{H}$. The following observations and lemmas follow easily.

Observation 7.2.3. *If $\overrightarrow{G}$ is not Eulerian then any correction subgraph $\overrightarrow{H}$ of $\overrightarrow{G}$ contains a spring-drain path.* □

Observation 7.2.4. *Let $\overrightarrow{H}$ be a correction subgraph of $\overrightarrow{G}$ and let $\overrightarrow{P}$ be a spring-drain path in $\overrightarrow{H}$. Define $\overrightarrow{H} \setminus \overrightarrow{P}$ to be the graph obtained from $\overrightarrow{H}$ by removing all the edges of $\overrightarrow{P}$, that is, $\overrightarrow{E} \setminus \overrightarrow{P} \stackrel{\text{def}}{=} (V_H, \overrightarrow{E} \setminus \overrightarrow{P})$. Then $\overrightarrow{H} \setminus \overrightarrow{P}$ is a correction subgraph of $\overrightarrow{G}_{\overleftarrow{P}}$, the graph obtained from $\overrightarrow{H}$ by inverting all the edges of $\overrightarrow{P}$. Moreover, if $\overrightarrow{H}$ is acyclic then $\overrightarrow{H} \setminus \overrightarrow{P}$ is edgeless if and only if $\overrightarrow{G}_{\overleftarrow{P}}$ is Eulerian.* □

Lemma 7.2.5. *If $\overrightarrow{G}$ is not Eulerian then any acyclic correction subgraph $\overrightarrow{H}$ of $\overrightarrow{G}$ is a union of $p = \frac{1}{4}\sum_{u\in V}|\mathrm{ib}(u)|$ edge-disjoint spring-drain paths. Moreover, every decomposition of $\overrightarrow{H}$ into edge-disjoint spring-drain paths has exactly p paths. In particular, removing any spring-drain path from $\overrightarrow{H}$ results in a graph that is a collection of $p-1$ edge-disjoint spring-drain paths.*

Proof. From Observations 7.2.3 and 7.2.4, it is clear that $\overrightarrow{H}$ may be decomposed into edge-disjoint spring-drain paths. Note that when removing a spring-drain path, we reduce the sum $\sum_{u\in V}|\mathrm{ib}(u)|$ by exactly four (as we reduce the absolute value of the imbalance of both the spring and the drain by two). Therefore, every decomposition of $\overrightarrow{H}$ into disjoint spring-drain paths contains $p = \frac{1}{4}\sum_{u\in V}|\mathrm{ib}(u)|$ paths. □

Lemma 7.2.6. *Let $\overrightarrow{G}$ be an orientation of G and let $\overrightarrow{H}$ be an acyclic correction subgraph of $\overrightarrow{G}$ which is a union of p disjoint spring-drain paths. Let S be the set of springs in $\overrightarrow{H}$. Then $\sum_{u\in S}\mathrm{ib}(u) = 2p$.*

Proof. By Lemma 7.2.1, for any spring $u \in S$, the number of spring-drain paths starting at u is $\mathrm{ib}(u)/2$. Thus $\sum_{u\in S}\mathrm{ib}(u)/2 = p$. □

Given some positive number p, if every correction subgraph of an orientation $\overrightarrow{G}$ is a union of at least p disjoint spring-drain paths, then we say that $\overrightarrow{G}$ is *p-far* from being Eulerian. An algorithm is be called a *p-test* for being Eulerian if it can distinguish between Eulerian orientations and p-far orientations. Namely, the algorithm should accept an Eulerian orientation with probability at least $2/3$ and reject a p-far orientation with probability at least $2/3$. Similarly to ε-tests, if a p-test accepts every Eulerian orientation with probability 1 then it is called *1-sided*, and otherwise it is called *2-sided*.

7.3 β-correction subgraphs and (p,β)-tests

Given $\beta > 0$, we say that a vertex v is *β-small* if $\deg(v) \le \beta$ and *β-big* if $\deg(v) > \beta$. An orientation $\overrightarrow{G}$ is called *β-Eulerian* if all the β-small vertices in V are balanced in $\overrightarrow{G}$. Note that for $\beta \ge \Delta$, $\overrightarrow{G}$ is β-Eulerian if and only if $\overrightarrow{G}$ is Eulerian. A directed subgraph $\overrightarrow{H} = (V_H, \overrightarrow{E_H})$ of $\overrightarrow{G}$ is a *β-correction subgraph of $\overrightarrow{G}$* if:

1. $\overrightarrow{G}_{\overleftarrow{H}}$ is β-Eulerian.

2. For every $v \in V$ we have: $|\mathrm{ib}_{\overrightarrow{G}_{\overrightarrow{H}}}(v)| \leq |\mathrm{ib}_{\overrightarrow{G}}(v)|$, and if v is a spring (resp. drain) in $\overrightarrow{H}$ then it is either balanced or a spring (resp. drain) in $\overrightarrow{G}_{\overrightarrow{H}}$.

That is, $\overrightarrow{H}$ fixes the balance of all the β-small vertices without increasing or changing the sign of the imbalance of β-big vertices.

A directed path $\overrightarrow{P} = \langle u_0, \ldots, u_k \rangle$ in an orientation $\overrightarrow{G}$ is called a *β-spring-drain path* if $\overrightarrow{P}$ is a spring-drain path and at least one of u_0 and u_k is β-small.

Observation 7.3.1. *Every orientation $\overrightarrow{G}$ which is not β-Eulerian has a β-correction subgraph which is a union of edge-disjoint β-spring-drain paths.* □

For $0 \leq \varepsilon \leq 1$ and $\beta > 0$, we say that an orientation $\overrightarrow{G}$ is *(ε, β)-amendable* if there exists a β-correction subgraph $\overrightarrow{H} = (V_H, \overrightarrow{E_H})$ of $\overrightarrow{G}$ with $|\overrightarrow{E_H}| \leq \varepsilon m$. Otherwise we say that $\overrightarrow{G}$ is *(ε, β)-unamendable*.

For $p, \beta > 0$, we say that an orientation $\overrightarrow{G}$ is *p-close to being β-Eulerian* if there exists a β-correction subgraph $\overrightarrow{H}$ of $\overrightarrow{G}$ that is a union of at most p edge-disjoint β-spring-drain paths. Otherwise, we say that $\overrightarrow{G}$ is *p-far from being β-Eulerian*. Note that in fact the requirement from the correction subgraph to be composed of β-spring-drain paths only is not restricting in the definition of being p-far.

One can show that all the lemmas and observations that we have proved in Section 7.2 for correction subgraphs and spring-drain paths can be adapted to β-correction subgraphs and β-spring-drain paths. In particular:

Lemma 7.3.2. *If $\overrightarrow{G}$ is not β-Eulerian then any edge-minimal (acyclic) β-correction subgraph $\overrightarrow{H}$ of $\overrightarrow{G}$ is a union of at least $\frac{1}{4}\sum_{u \in V, \deg(u) \leq \beta} |\mathrm{ib}(u)|$ and at most $\frac{1}{2}\sum_{u \in V, \deg(u) \leq \beta} |\mathrm{ib}(u)|$ edge-disjoint β-spring-drain paths. Hence, if $\overrightarrow{G}$ is p-far from being β-Eulerian then*

$$\sum_{u \in V, \deg(u) \leq \beta} |\mathrm{ib}(u)| > 2p.$$

Proof. One can show that any acyclic β-correction subgraph is a union of edge-disjoint β-spring-drain paths, using similar techniques to those that we used in Section 7.2, as well as the fact that a β-correction subgraph may not increase or change the sign of the imbalance of β-big vertices. To bound the number of β-spring-drain paths in any β-correction subgraph $\overrightarrow{H}$ of an orientation $\overrightarrow{G}$, note that when we remove a β-spring-drain path from $\overrightarrow{H}$, we reduce the sum $\sum_{u \in V, \deg(u) \leq \beta} |\mathrm{ib}(u)|$ by either two or four (as we reduce the absolute value of the imbalance of both the spring and the drain by two, and at least one of them is β-small). Therefore, every

decomposition of $\overrightarrow{H}$ into disjoint β-spring-drain paths contains p paths for some p between $\frac{1}{4}\sum_{u\in V,\deg(u)\leq\beta}|\mathrm{ib}(u)|$ and $\frac{1}{2}\sum_{u\in V,\deg(u)\leq\beta}|\mathrm{ib}(u)|$. $\square$

Given $p,\beta>0$, an algorithm is called a (p,β)-*test for being Eulerian* for some positive number p if it can distinguish between β-Eulerian orientations and orientations that are p-far from being β-Eulerian. Namely, the algorithm should accept a β-Eulerian orientation with probability at least $2/3$ and reject an orientation that is p-far from being β-Eulerian with probability at least $2/3$. As usual, a (p,β)-test is said to be *1-sided* if it accepts every β-Eulerian orientation with probability 1. Otherwise, the test is said to be *2-sided*.

7.4 A linear lower bound for 1-sided tests

In this section we prove that there exists no sub-linear 1-sided test for Eulerian orientations of general graphs.

Consider an algorithm that tests an orientation $\overrightarrow{G}$ of G. At a given moment, we represent the edges that the algorithm has queried so far by a directed *knowledge graph* $\overrightarrow{H}=(V,\overrightarrow{E_H})$, where $\overrightarrow{E_H}\subseteq\overrightarrow{E}$. We say that a cut $M=(U,V\setminus U)$ of G is *valid* with respect to a knowledge graph $\overrightarrow{H}$ if

$$|\overrightarrow{E_H}(U,V\setminus U)| \;\leq\; \frac{1}{2}|E(U,V\setminus U)| \quad\text{and}\quad |\overrightarrow{E_H}(V\setminus U,U)| \;\leq\; \frac{1}{2}|E(U,V\setminus U)|.$$

Otherwise, M is called *invalid*. Clearly, if $\overrightarrow{G}$ is Eulerian, then every knowledge graph $\overrightarrow{H}$ of $\overrightarrow{G}$ contains only valid cuts. We show that any 1-sided test for being Eulerian must obtain a knowledge graph that contains some invalid cut in order to reject $\overrightarrow{G}$.

We say that a (valid) cut $M=(U,V\setminus U)$ of G is *restricting* with respect to $\overrightarrow{H}$ if

$$|\overrightarrow{E_H}(U,V\setminus U)| \;=\; \frac{1}{2}|E(U,V\setminus U)| \quad\text{or}\quad |\overrightarrow{E_H}(V\setminus U,U)| \;=\; \frac{1}{2}|E(U,V\setminus U)|.$$

Note that, given that $\overrightarrow{G}$ is Eulerian, a restricting cut with respect to $\overrightarrow{H}$ forces the orientations of all the unqueried edges in the cut. We say that two restricting cuts M_1 and M_2 are *conflicting* (with respect to a knowledge graph $\overrightarrow{H}$) if they force contrasting orientations of at least one unqueried edge.

Lemma 7.4.1. *Let $\overrightarrow{H}$ be a knowledge graph of $\overrightarrow{G}$ and suppose that all the cuts in G are valid with respect to $\overrightarrow{H}$. Then there are no conflicting cuts with respect to $\overrightarrow{H}$.*

Proof. Assume, on the contrary, that $M_1 = (V_1, V \setminus V_1)$ and $M_2 = (V_2, V \setminus V_2)$ are conflicting with respect to $\overrightarrow{H}$. That is, there exists an edge $\{u, w\} \in E$, that was not queried and hence is not oriented in $\overrightarrow{H}$, which is forced to have contrasting orientations by M_1 and M_2. Without loss of generality, assume that $u \in V_1 \setminus V_2,\ w \in V_2 \setminus V_1$, and

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| \ = \ \frac{1}{2} \cdot |E(V_1,\ V \setminus V_1)|, \tag{7.1}$$

$$|\overrightarrow{E_H}(V_2,\ V \setminus V_2)| \ = \ \frac{1}{2} \cdot |E(V_2,\ V \setminus V_2)|. \tag{7.2}$$

Thus, M_1 forces e to be oriented from w to u, whereas M_2 forces e to be oriented from u to w.

Recall now that all the cuts in G are valid with respect to $\overrightarrow{H}$. Consider the cuts $(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))$ and $(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))$. We have

$$|\overrightarrow{E_H}(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \ \leq \ \frac{1}{2} \cdot |E(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \tag{7.3}$$

and

$$|\overrightarrow{E_H}(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \ \leq \ \frac{1}{2} \cdot |E(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \tag{7.4}$$

since these cuts are valid. Note that

$$|E(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \ + \ |E(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \tag{7.5}$$

$$= |E(V_1,\ V \setminus V_1)| + |E(V_2,\ V \setminus V_2)| - 2 \cdot |E(V_1 \setminus V_2,\ V_2 \setminus V_1)|$$

and

$$|\overrightarrow{E_H}(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \ + \ |\overrightarrow{E_H}(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| = \tag{7.6}$$

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| + |\overrightarrow{E_H}(V_2,\ V \setminus V_2)| - |\overrightarrow{E_H}(V_1 \setminus V_2,\ V_2 \setminus V_1)| - |\overrightarrow{E_H}(V_2 \setminus V_1,\ V_1 \setminus V_2)|.$$

Summing Equation (7.1) with Equation (7.2) yields

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| + |\overrightarrow{E_H}(V_2,\ V \setminus V_2)| = \ \frac{1}{2} \left(|E(V_1,\ V \setminus V_1)| \ + \ |E(V_2,\ V \setminus V_2)| \right). \tag{7.7}$$

Summing Inequality (7.3) with Inequality (7.4) yields

$$|\overrightarrow{E_H}(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \ +\ |\overrightarrow{E_H}(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))| \leq \tag{7.8}$$

$$\frac{1}{2}\left(|E(V_1 \cap V_2,\ V \setminus (V_1 \cap V_2))| \ +\ |E(V_1 \cup V_2,\ V \setminus (V_1 \cup V_2))|\right).$$

Substituting Equations (7.5) and (7.6) in Inequality (7.8) we obtain:

$$|\overrightarrow{E_H}(V_1,\ V \setminus V_1)| + |\overrightarrow{E_H}(V_2,\ V \setminus V_2)| - |\overrightarrow{E_H}(V_1 \setminus V_2,\ V_2 \setminus V_1)| - |\overrightarrow{E_H}(V_2 \setminus V_1,\ V_1 \setminus V_2)| \leq$$

$$\frac{1}{2}\left(|E(V_1,\ V \setminus V_1)| \ +\ |E(V_2,\ V \setminus V_2)|\right) - |E(V_1 \setminus V_2,\ V_2 \setminus V_1)|.$$

Now, from Equation (7.7) we have:

$$|\overrightarrow{E_H}(V_1 \setminus V_2,\ V_2 \setminus V_1)| \ +\ |\overrightarrow{E_H}(V_2 \setminus V_1,\ V_1 \setminus V_2)| \ \geq\ |E(V_1 \setminus V_2,\ V_2 \setminus V_1)|.$$

That is, all the edges in $E(V_1 \setminus V_2,\ V_2 \setminus V_1)$ are oriented in $\overrightarrow{H}$. This is a contradiction to our assumption that $\{u, w\} \in E(V_1 \setminus V_2,\ V_2 \setminus V_1)$ was not yet oriented. □

Lemma 7.4.2. *Suppose that $\overrightarrow{H}$ is a knowledge graph that does not contain invalid cuts. Then $\overrightarrow{H}$ is extensible to an Eulerian orientation $\overrightarrow{G} = (V, \overrightarrow{E_G})$ of G. That is, $\overrightarrow{E_H} \subseteq \overrightarrow{E_G}$.*

Proof. We orient unoriented edges in the following manner. If there exists a restricting cut with unoriented edges, we orient one of them as obliged by the cut. According to Lemma 7.4.1, this will not invalidate any of the other cuts in the graph, and so we may continue. If there are no restricting cuts in the graph, we arbitrarily orient one unoriented edge and repeat (and this cannot violate any cut in the graph since there were no restricting cuts). Eventually, after orienting all the edges, we receive a complete orientation of G whose cuts are all valid, and thus it is Eulerian. □

Theorem 7.4.3. *There exists an infinite family of graphs for which every 1-sided test for being Eulerian must use $\Omega(m)$ queries.*

Proof. For every even n, let $G_n \stackrel{\text{def}}{=} K_{2,n-2}$, namely, the graph with a set of vertices $V = \{v_1, \ldots, v_n\}$ and a set of edges $E = \{\{v_i, v_j\} \mid i \in \{1, 2\},\ j \in \{3, \ldots, n\}\}$. Clearly, G_n is Eulerian and $n = \Omega(m)$.

Consider the orientation $\overrightarrow{G}_n$ of G_n in which all the edges incident with v_1 are outgoing and all the edges incident with v_2 are incoming. Clearly, $\overrightarrow{G}_n$ is $\frac{1}{2}$-far from being Eulerian. According to Lemma 7.4.2, every 1-sided test must query at least half of the edges in some unbalanced cut (because otherwise it would clearly not obtain an invalid cut in the knowledge graph). However, one can easily see that every cut which does not separate v_1 and v_2 is balanced, while every cut which separates v_1 and v_2 is of size $n - 2 = \Omega(m)$. $\square$

Chapter 8

Upper Bounds

8.1 Generic tests

In this section we present one 1-sided p-test and two 2-sided p-tests for being Eulerian. Namely, our tests distinguish with high probability between the case where $\overrightarrow{G}$ is Eulerian and the case where $\overrightarrow{G}$ is p-far from being Eulerian (see Section 7.2). In later sections we devise several lower bounds on p for every orientation $\overrightarrow{G}$ that is ε-far from being Eulerian, thus obtaining corresponding upper bounds on the tests below. In fact, the 1-sided and 2-sided tests that we give in Subsection 8.1.2 are (p, β)-tests (see Section 7.3), which are, in particular, p-tests when $\beta = \Delta$. We will use these tests also for $\beta < \Delta$ in Section 8.4.

8.1.1 A 2-sided p-test

We give a simple 2-sided p-test that is independent of the maximum degree Δ. This p-test will yield efficient ε-tests for dense graphs (Section 8.2) and expanders (Section 8.3). To simplify notation, we denote $\delta \stackrel{\text{def}}{=} \frac{p}{4m}$.

Algorithm 8.1.1. *SIMPLE-2($\overrightarrow{G}, p$):*

1. *Repeat $\frac{4}{\delta}$ times independently:*
    - *Select an edge $e \in E$ uniformly and query it. Denote the start vertex of e in $\overrightarrow{E}$ by u and the end vertex of e in $\overrightarrow{E}$ by v.*
    - *Query $\frac{16\ln(12/\delta)}{\delta^2}$ edges incident with u uniformly and independently and reject if the sample contains at least $(1+\delta)\frac{8\ln(12/\delta)}{\delta^2}$ outgoing edges.*

2. Accept if the input was not rejected earlier.

Lemma 8.1.2. *SIMPLE-2 is a 2-sided p-test for being Eulerian with query complexity* $\widetilde{O}\left(\frac{1}{\delta^3}\right) = \widetilde{O}\left(\frac{m^3}{p^3}\right)$.

Proof. The query complexity is clearly as stated. To prove the correctness of our algorithm, suppose first that $\overrightarrow{G}$ is Eulerian. For every vertex $u \in V$, the expected number of outgoing edges in the sample of u's incident edges is $\frac{8\ln(12/\delta)}{\delta^2}$. Applying Chernoff's upper tail bound, the probability of having at least $(1+\delta)\frac{8\ln(12/\delta)}{\delta^2}$ outgoing edges in a sample is at most

$$\left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\frac{8\ln(12/\delta)}{\delta^2}}. \tag{8.1}$$

From the Taylor expansion, we have

$$\ln(1+\delta) = \delta - \frac{\delta^2}{2} + \frac{\delta^3}{3} - \dots$$

and

$$\delta \cdot \ln(1+\delta) = \delta^2 - \frac{\delta^3}{2} + \frac{\delta^4}{3} - \dots.$$

Summing the two series we obtain that $(1+\delta)\ln(1+\delta) > \delta + \frac{\delta^2}{2} - \frac{\delta^3}{6} > \delta + \frac{\delta^2}{8}$. Thus $(1+\delta)\ln(1+\delta) - \delta > \frac{\delta^2}{8}$ and therefore $\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} < e^{-\delta^2/8}$. Substituting in Equation (8.1), we obtain that the probability of sampling $(1+\delta)\frac{8\ln(12/\delta)}{\delta^2}$ outgoing edges for a balanced vertex is at most $\frac{\delta}{12}$. Since we sample $\frac{4}{\delta}$ vertices, the probability of rejecting an Eulerian orientation is at most $\frac{1}{3}$.

Assume now that $\overrightarrow{G}$ is p-far from being Eulerian. A vertex $u \in V$ will be called a *δ-spring* in $\overrightarrow{G}$ if $\mathrm{ib}(u) > 3\delta \cdot \deg(u)$. Let S_δ be the set of all δ-springs in $\overrightarrow{G}$. Note that $S_\delta \subseteq S$, where S is the set of all springs in $\overrightarrow{G}$. We have

$$\sum_{u \in S} \mathrm{ib}(u) \leq \sum_{u \in S_\delta} \deg(u) + 3\delta \cdot \sum_{u \in S \setminus S_\delta} \deg(u).$$

Let $y = \sum_{u \in S_\delta} \deg(u)$. Since $\sum_{u \in S} \deg(u) < 2m$, we obtain

$$\sum_{u \in S} \mathrm{ib}(u) < y + 3\delta \cdot (2m - y) = (1 - 3\delta)y + 6\delta m.$$

However, from Lemma 7.2.6, we have $\sum_{u \in S} \mathrm{ib}(u) = 2p = 8\delta m$, and therefore,

$$y > \frac{2\delta m}{1-3\delta} > 2\delta m.$$

Thus, $\sum_{u \in S_\delta} \deg(u) > 2\delta m$, and since for every $u \in S_\delta$ we have $\deg_{out}(u) > \frac{1}{2}\deg(u)$, there exist at least δm edges $(u,v) \in \overrightarrow{E}$ such that u is a δ-spring. As SIMPLE-2 samples $\frac{4}{\delta}$ edges $(u,v) \in \overrightarrow{E}$ uniformly and independently, the probability of not sampling any edge (u,v) such that u is a δ-spring is at most $(1-\delta)^{\frac{4}{\delta}} < e^{-4} < 0.03$.

Suppose now that the algorithm has sampled an edge (u,v) such that u is a δ-spring. Then $\overrightarrow{G}$ will be rejected unless fewer than $(1+\delta)\frac{8\ln(12/\delta)}{\delta^2}$ of the queried edges (u,w) are outgoing. Since $\mathrm{ib}(u) > 3\delta \cdot \deg(u)$, the expected number of outgoing edges is at least $(1+3\delta/2)\frac{8\ln(12/\delta)}{\delta^2}$. Note that $(1+\delta)\frac{8\ln(12/\delta)}{\delta^2} < (1-\frac{1}{4}\delta)(1+\frac{3}{2}\delta)\frac{8\ln(12/\delta)}{\delta^2}$. Therefore, by the Chernoff bound, the probability of sampling fewer than $(1+\delta)\frac{8\ln(12/\delta)}{\delta^2}$ outgoing edges is at most

$$\exp\left(-\frac{\delta^2}{32}\left(1+\frac{3\delta}{2}\right)\frac{8\ln(12/\delta)}{\delta^2}\right) < \exp\left(-\frac{\ln(12/\delta)}{4}\right).$$

Note that $\frac{\ln(12/\delta)}{4} > \ln(10/3)$, and therefore, the probability of not detecting that u is a δ-spring is at most 0.3. We thus conclude that if $\overrightarrow{G}$ is p-far from being Eulerian, then SIMPLE-2 accepts $\overrightarrow{G}$ with probability smaller than $\frac{1}{3}$, which completes the proof of the theorem. $\square$

8.1.2 (p,β)-tests

We now give a simple 1-sided (p,β)-test, which has a better query complexity than SIMPLE-2 for $\Delta \ll \frac{m^2}{p^2}\ln(\frac{m}{p})$.

Algorithm 8.1.3. *GENERIC-1($\overrightarrow{G}, p, \beta$):*

1. *Repeat $\frac{\ln 3 \cdot m}{p}$ times independently:*
   - *Select an edge $e \in E$ uniformly and query it. Denote the start vertex of e in $\overrightarrow{E}$ by u and the end vertex of e in $\overrightarrow{E}$ by v.*
   - *If $\deg(u) \leq \beta$ then: Query all the edges $\{u,w\} \in E$ and reject if u is unbalanced, namely, if $\mathrm{ib}(u) \neq 0$.*
2. *Repeat $\frac{\ln 3 \cdot m}{p}$ times independently:*

- *Select an edge $e \in E$ uniformly and query it. Denote the start vertex of e in $\overrightarrow{E}$ by u and the end vertex of e in $\overrightarrow{E}$ by v.*
- *If $\deg(v) \leq \beta$ then: Query all the edges $\{w, v\} \in E$ and reject if v is unbalanced, namely, if $\mathrm{ib}(v) \neq 0$.*

3. Accept if the input was not rejected by the above.

Lemma 8.1.4. *GENERIC-1 is a 1-sided (p, β)-test for being Eulerian with query complexity $O\left(\frac{\beta m}{p}\right)$. In particular, for $\beta = \Delta$, GENERIC-1 is a 1-sided p-test with query complexity $O\left(\frac{\Delta m}{p}\right)$.*

Proof. The query complexity is clearly as stated. Since Algorithm 8.1.3 rejects only when an unbalanced β-small vertex is discovered, it always accepts a β-Eulerian $\overrightarrow{G}$. Suppose now that $\overrightarrow{G}$ is p-far from being β-Eulerian. Clearly, to reject $\overrightarrow{G}$, it suffices to sample one edge $e = (u, v) \in \overrightarrow{E}$ such that u is a spring in Step 1 or an edge $e = (u, v) \in \overrightarrow{E}$ such that v is a drain in Step 2. By Lemma 7.3.2, E contains at least p edges of at least one of these two kinds of edges, and so their fraction is at least p/m. We thus conclude that the probability of accepting $\overrightarrow{G}$ is at most

$$\left(1 - \frac{p}{m}\right)^{\frac{\ln 3 \ m}{p}} < \frac{1}{3}.$$

□

We conclude this section with a 2-sided (p, β)-test, which gives better query complexity than GENERIC-1 for $\beta \gg \log^2 m$ and better query complexity than SIMPLE-2 for $p \ll \frac{m}{\sqrt{\beta}}$. The main idea of the algorithm is to perform roughly $O((\log \beta)^2)$ testing stages, each designed to detect unbalanced vertices whose degree and imbalance lie in a certain interval. We show that with high probability, a β-Eulerian orientation $\overrightarrow{G}$ is accepted by all the testing stages, while an orientation that is p-far from being β-Eulerian is rejected by at least one of them.

Algorithm 8.1.5. *MULTISTAGE-2($\overrightarrow{G}, p, \beta$):*

For $i = 1, \ldots, \lceil \log \beta \rceil - 1$, do:

1. Let $V_i \stackrel{\text{def}}{=} \{u \in V \mid \deg(u) \in [2^i, 2^{i+1})\}$ and $n_i \stackrel{\text{def}}{=} |V_i|$.

2. Let $j = \lceil i/2 \rceil$. If $2^j \cdot n_i > \frac{2p}{(\log \beta)^2}$ then:

- *Sample* $x_{ij} = \frac{\ln 12 \ (\log\beta)^2 \ 2^{j+1} \ n_i}{2p}$ *vertices in* V_i *uniformly and independently.*
- *For every sampled vertex* u*, query all the edges incident with* u*, and reject if* u *is unbalanced.*

3. *For every* $j \in \{\lceil i/2 \rceil + 1, \ldots, i-1\}$ *such that* $2^j \cdot n_i > \frac{2p}{(\log\beta)^2}$ *do:*
   - *Sample* $x_{ij} = \frac{\ln 12 \ (\log\beta)^2 \ 2^{j+1} \ n_i}{2p}$ *vertices in* V_i *uniformly and independently.*
   - *For every sampled vertex* u*, query* $q_{ij} = 256 \cdot \ln(\ 6(\log\beta)^2 \ x_{ij}) \cdot 2^{2(i-j)}$ *edges adjacent to* u*, uniformly and independently, and reject if the absolute difference between the number of incoming and outgoing edges in the sample is at least* $\frac{q_{ij}}{4 \cdot 2^{i-j}}$.

Accept if the input was not rejected earlier.

Lemma 8.1.6. *MULTISTAGE-2 is a 2-sided* (p, β)*-test for being Eulerian with query complexity* $\widetilde{O}\left(\frac{\sqrt{\beta} \ m}{p}\right)$*. In particular, for* $\beta = \Delta$*, MULTISTAGE-2 is a 2-sided* p*-test for being Eulerian with query complexity* $\widetilde{O}\left(\frac{\sqrt{\Delta} \ m}{p}\right)$.

Proof. First, we compute the asymptotic query complexity of Algorithm 8.1.5. Since

$$x_{ij} = O\left(\frac{(\log\beta)^2 \cdot 2^j \cdot n_i}{p}\right)$$

and

$$q_{ij} = O\left(\log\left(\frac{\log\beta \cdot 2^j \cdot n_i}{p}\right) \cdot 2^{2(i-j)}\right) = O\left(\log\left(\frac{\log\beta \cdot m}{p}\right) \cdot 2^{2(i-j)}\right),$$

the total query complexity is at most

$$\sum_{i=1}^{\log\beta} \left(x_{i\lceil i/2 \rceil} 2^{\lceil i/2 \rceil + 1} + \sum_{j=\lceil i/2 \rceil + 1}^{i-1} x_{ij} \cdot q_{ij} \right)$$

$$= O\left(\log\left(\frac{\log\beta \cdot m}{p}\right) \frac{(\log\beta)^2}{p} \sum_{i=1}^{\log\beta} 2^{2i} \cdot n_i \sum_{j=\lceil i/2 \rceil}^{i-1} \frac{1}{2^j}\right)$$

$$= O\left(\log\left(\frac{\log\beta \cdot m}{p}\right) \frac{(\log\beta)^2}{p} \sum_{i=1}^{\log\beta} 2^{3i/2} \cdot n_i\right)$$

$$= O\left(\log\left(\frac{\log\beta\cdot m}{p}\right)\frac{(\log\beta)^2}{p}\ 2^{\log\beta/2}\sum_{i=1}^{\log\beta}\ 2^i\cdot n_i\right)$$
$$= O\left(\log\left(\frac{\log\beta\cdot m}{p}\right)\frac{(\log\beta)^2}{p}\sqrt{\beta}m\right).$$

Now suppose that $\overrightarrow{G}$ is β-Eulerian. Then $\overrightarrow{G}$ can only be rejected in Step 3, where we randomly sample q_{ij} edges incident with a (β-small) vertex $u \in V_i$. Since u is balanced, the expected number of incoming edges in the sample is $\frac{q_{ij}}{2}$. By the Chernoff bound, the probability of sampling fewer than $(1-\frac{1}{4\cdot 2^{i-j}})\frac{q_{ij}}{2}$ incoming edges is at most $\exp\left(-\frac{1}{2^{2(i-j)}}\cdot\frac{q_{ij}}{64}\right) < \frac{1}{6(\log\beta)^2 x_{ij}}$. Similarly, the probability of sampling fewer than $(1-\frac{1}{4\cdot 2^{i-j}})\frac{q_{ij}}{2}$ outgoing edges is at most $\frac{1}{6(\log\beta)^2 x_{ij}}$. Since for every pair (i,j) we sample x_{ij} vertices, and since there are less than $(\log\beta)^2$ pairs, the total probability of rejecting $\overrightarrow{G}$ is at most $\frac{1}{3}$.

Suppose now that $\overrightarrow{G}$ is p-far from being β-Eulerian. From Lemma 7.3.2, we have

$$\sum_{u\in V,\deg(u)\leq\beta}|\mathrm{ib}(u)| > 2p.$$

We define a partition of the unbalanced β-small vertices in $\overrightarrow{G}$ as follows. For every $1 \leq i \leq \lceil\log\beta\rceil - 1$ and for every $1 \leq j \leq i$, let

$$V_{ij} \stackrel{\text{def}}{=} \{u \in V_i \mid |\mathrm{ib}(u)| \in [2^j, 2^{j+1})\ \}.$$

Then there exist i_0 and j_0 such that

$$\sum_{u\in V_{i_0j_0}}|\mathrm{ib}(u)| > \frac{2p}{(\log\beta)^2}. \tag{8.2}$$

As $|\mathrm{ib}(u)| < 2^{j_0+1}$ for every $u \in V_{i_0j_0}$, we have

$$|V_{i_0j_0}| > \frac{2p}{(\log\beta)^2\cdot 2^{j_0+1}}.$$

Recall that $|V_{i_0}| = n_{i_0}$. Hence, when we sample $x_{i_0j_0}$ vertices in Step 2 or in Step 3, the probability of not sampling any vertex $u \in V_{i_0j_0}$ is smaller than

$$\left(1-\frac{2p}{(\log\beta)^2\cdot 2^{j_0+1}\cdot n_{i_0}}\right)^{x_{i_0j_0}} < \exp\left(-\frac{2p}{(\log\beta)^2\cdot 2^{j_0+1}\cdot n_{i_0}}\cdot x_{i_0j_0}\right) = \frac{1}{12}.$$

Assume from now on that the algorithm has sampled a vertex $u \in V_{i_0 j_0}$. If $j_0 \leq \lceil i_0/2 \rceil$ then all the edges incident with u are queried (Step 2), and since u is unbalanced, $\overrightarrow{G}$ is now rejected with probability 1. Otherwise, $q_{i_0 j_0}$ edges incident with u are queried independently in Step 3. Since $\deg(u) < 2^{i_0+1}$ and $|\mathrm{ib}(u)| \geq 2^{j_0}$, either the expected number of incoming edges in the sample is at least $\left(1 + \frac{1}{2 \cdot 2^{i_0 - j_0}}\right) \frac{q_{i_0 j_0}}{2}$ or the expected number of outgoing edges in the sample is at least $\left(1 + \frac{1}{2 \cdot 2^{i_0 - j_0}}\right) \frac{q_{i_0 j_0}}{2}$. To accept the input, we must sample fewer than $\left(1 + \frac{1}{4 \cdot 2^{i_0 - j_0}}\right) \frac{q_{i_0 j_0}}{2} < \left(1 - \frac{1}{8 \cdot 2^{i_0 - j_0}}\right) \cdot \left(1 + \frac{1}{2 \cdot 2^{i_0 - j_0}}\right) \frac{q_{i_0 j_0}}{2}$ edges in the majority direction. By the Chernoff bound, the probability of doing so is at most

$$\exp\left(-\frac{q_{i_0 j_0}}{256 \cdot 2^{2(i_0 - j_0)}}\right) = \exp\left(-\ln(6(\log \beta)^2 \; x_{i_0 j_0})\right) = \frac{1}{6(\log \beta)^2 \; x_{i_0 j_0}}.$$

Note that from Inequality (8.2) we must have $2^{j_0+1} \cdot n_{i_0} > \frac{4p}{(\log \beta)^2}$ and hence $x_{i_0 j_0} > \ln 12$. In addition, we may assume that $\beta \geq 2$, since otherwise $\overrightarrow{G}$ is trivially β-Eulerian. It thus follows that the probability of not rejecting the input when sampling u's edges is smaller than $1/4$. We conclude that the probability of accepting an orientation that is p-far from being β-Eulerian is at most $1/3$. $\square$

We note that the query complexity bound of MULTISTAGE-2 can be made tighter for some graphs, as the algorithm skips pairs (i, j) where $2^j \cdot n_i \leq \frac{2p}{(\log \beta)^2}$ (and thus i and j cannot be the i_0 and j_0 which satisfy Equation (8.2)). In particular, for regular graphs, only one value of i is relevant for testing, which eliminates the square over $\log \beta$ in the query complexity. However, this does not change the power of β or the dependency on p and m.

8.2 Testing graphs with high average degree

In this section we obtain a general lower bound for the required number p of correction paths as a function of $d = m/n$. This, together with the analysis of the previous section, will provide us with an efficient test for dense graphs.

Lemma 8.2.1. *Suppose that $\overrightarrow{G}$ is not Eulerian and that $\overrightarrow{H}$ is an acyclic correction subgraph of $\overrightarrow{G}$ which is a union of p edge-disjoint spring-drain paths. Then $\overrightarrow{H}$ contains a spring-drain path of length smaller than $\frac{n}{\sqrt{p}}$.*

Proof. Consider a Breadth-First Search (BFS) traversal of $\overrightarrow{H}$ starting at the set of springs, S. We define a partition of V_H into levels $L_0, \ldots, L_t$ for some $t > 0$ as

follows. Let $L_0 = S$. Now, for any $i > 0$, while $\bigcup_{j=0}^{i-1} L_j \neq V_H$, define

$$L_i \stackrel{\text{def}}{=} \{v \in V_H \setminus \bigcup_{j=0}^{i-1} L_j \mid \text{there exists } u \in L_{i-1} \text{ such that } (u,v) \in \overrightarrow{E_H}\}.$$

Note that if $v \in L_i$ then $\overrightarrow{H}$ contains a path of length i from some spring to v. Let ℓ be the minimum index of a level L_i which contains a drain. We prove the claim by showing that $\ell < \frac{n}{\sqrt{p}}$.

For every $i = 0, \ldots, t$ we let $n_i = |L_i|$. Recall that there are p edge-disjoint spring-drain paths in $\overrightarrow{H}$. We first show that for every $0 \leq i < \ell$ we have $n_{i+1} \geq p/n_i$. Consider the level L_i for some $0 \leq i < \ell$. Since there are no drains in the levels $L_0, \ldots, L_i$, there exist at least p edges from L_i to L_{i+1}. Therefore, there exists a vertex $v \in L_i$ which has at least p/n_i neighbors in L_{i+1}. Hence, $n_{i+1} \geq p/n_i$.

Summing over all $i = 0, \ldots, \ell - 1$ we obtain

$$\sum_{i=0}^{\ell-1} n_{i+1} \geq p \cdot \sum_{i=0}^{\ell-1} \frac{1}{n_i},$$

and so

$$p \leq \frac{\sum_{i=0}^{\ell-1} n_{i+1}}{\sum_{i=0}^{\ell-1} \frac{1}{n_i}}.$$

Now, for a given $\sum_{i=0}^{\ell-1} n_i$, the minimum value of $\sum_{i=0}^{\ell-1} \frac{1}{n_i}$ is reached when $n_i = \sum_{i=0}^{\ell-1} n_i/\ell$ for every $i = 0, \ldots, \ell - 1$. Thus,

$$p \leq \frac{\sum_{i=0}^{\ell-1} n_{i+1}}{\ell^2 / \sum_{i=0}^{\ell-1} n_i} < n^2/\ell^2,$$

which proves the lemma. □

Lemma 8.2.2. *If $\overrightarrow{G}$ is ε-far from being Eulerian then it is p-far from being Eulerian for $p > \varepsilon^2 d^2/4$.*

Proof. Let $\overrightarrow{H_0}$ be an acyclic correction subgraph of $\overrightarrow{G}$. In each step $j \geq 1$, while $\overrightarrow{H_{j-1}}$ is not empty, we choose a shortest spring-drain path $\overrightarrow{P_{j-1}}$ in $\overrightarrow{H_{j-1}}$ and set $\overrightarrow{H_j} = \overrightarrow{H_{j-1}} \setminus \overrightarrow{P_{j-1}}$. By Lemma 7.2.5, $\overrightarrow{H_0}$ is a union of $p = \frac{1}{4}\sum_{u \in V} |\text{ib}(u)|$ edge-disjoint spring-drain paths, and moreover, every $\overrightarrow{H_j}$ is a union of $p - j$ disjoint spring-drain paths. Hence, the graphs $\overrightarrow{H_0}, \ldots, \overrightarrow{H_{p-1}}$ are non-empty. Furthermore, every subgraph

$\overrightarrow{H_j}$ is clearly acyclic, and hence by Observation 7.2.4, for $j = 0, \ldots, p-1$ $\overrightarrow{H_j}$ is a correction subgraph for some non-Eulerian orientation of G. Let ℓ_j be the length of $\overrightarrow{P_j}$ for $j = 0, \ldots, p-1$. Then by Lemma 8.2.1, we have $\ell_j < \frac{n}{\sqrt{p-j}}$. Summing over j, we obtain

$$\sum_{j=0}^{p-1} \ell_j < n \cdot \sum_{j=0}^{p-1} \frac{1}{\sqrt{p-j}} = n \sum_{j=1}^{p} \frac{1}{\sqrt{j}} = n \left(1 + \sum_{j=2}^{p} \frac{1}{\sqrt{j}}\right). \tag{8.3}$$

Since $f(x) = \frac{1}{\sqrt{x}}$ is monotone decreasing for every $x > 0$, we have

$$\frac{1}{\sqrt{j}} < \int_{x=j-1}^{j} \frac{dx}{\sqrt{x}}$$

for every $j \geq 1$, and therefore

$$\sum_{j=2}^{p} \frac{1}{\sqrt{j}} < \int_{x=1}^{p} \frac{dx}{\sqrt{x}} = 2\sqrt{p} - 2.$$

Substituting this in (8.3) we obtain

$$\sum_{j=0}^{p-1} \ell_j < 2\sqrt{p}n.$$

Note that $\sum_{j=0}^{p-1} \ell_j$ is the total number of edges in $\overrightarrow{H}$. As $\overrightarrow{G}$ is ε-far from being Eulerian, we have $\varepsilon m < \sum_{j=0}^{p-1} \ell_j$, and thus, $p > \varepsilon^2 m^2/4n^2 = \varepsilon^2 d^2/4$. $\square$

Substituting the lower bound for p of Lemma 8.2.2 in Lemmas 8.1.4, 8.1.2 and 8.1.6, we obtain the following theorem.

Theorem 8.2.3.

1. *SIMPLE-2($\overrightarrow{G}, \varepsilon^2 d^2/4$) is a 2-sided ε-test for being Eulerian with query complexity $\widetilde{O}\left(\frac{m^3}{\varepsilon^6 d^6}\right) = \widetilde{O}\left(\frac{n^3}{\varepsilon^6 d^3}\right)$.*

2. *GENERIC-1($\overrightarrow{G}, \varepsilon^2 d^2/4, \Delta$) is a 1-sided ε-test for being Eulerian with query complexity $O\left(\frac{\Delta m}{\varepsilon^2 d^2}\right) = O\left(\frac{\Delta n}{\varepsilon^2 d}\right)$.*

3. *MULTISTAGE-2($\overrightarrow{G}, \varepsilon^2 d^2/4, \Delta$) is a 2-sided ε-test for being Eulerian with query complexity $\widetilde{O}\left(\frac{\sqrt{\Delta} m}{\varepsilon^2 d^2}\right) = \widetilde{O}\left(\frac{\sqrt{\Delta} n}{\varepsilon^2 d}\right)$.*

SIMPLE-2 gives a sub-linear complexity for $d = \omega\left(\frac{\sqrt{n}\cdot(\log n)^{1/4}}{\varepsilon^{3/2}}\right)$, GENERIC-1 gives a sub-linear query complexity for $d = \omega\left(\frac{\sqrt{\Delta}}{\varepsilon}\right)$, and MULTISTAGE-2 gives a sub-linear complexity for $d = \omega\left(\frac{\Delta^{1/4}}{\varepsilon}\right)$. All of the tests yield their lowest query complexity relative to m when $m = \Theta(n^2)$ (i.e., $d = \Theta(n)$): $\widetilde{O}(1/\varepsilon^6)$ for SIMPLE-2, $O(n/\varepsilon^2)$ for GENERIC-1, and $\widetilde{O}(\sqrt{n}/\varepsilon^2)$ for MULTISTAGE-2.

8.3 Testing orientations of an expander graph

A graph $G = (V, E)$ is called an *α-expander* for some $\alpha > 0$ if it is connected and for every $U \subseteq V$ such that $0 < |E(U)| \leq m/2$ we have

$$\frac{|\partial U|}{|E(U)|} \geq \alpha.$$

Note that while the diameter of G is $O(\log_{(1+\alpha)} m)$, the "oriented-diameter" of $\overrightarrow{G}$ is not necessarily low, even if we assume that the orientation is Eulerian, as was shown by [6].

In the following, $\log_b^{(k)}(x)$ denotes the k-nested logarithm with base b of x, that is, $\log_b^{(1)}(x) \stackrel{\text{def}}{=} \log_b(x)$ and $\log_b^{(k+1)}(x) \stackrel{\text{def}}{=} \log_b(\log_b^{(k)}(x))$ for any natural $k \geq 1$.

Lemma 8.3.1. *Let G be an Eulerian α-expander and let $k \geq 1$ be a natural number such that $\log_{(1+\alpha/2)}^{(k-1)} m \geq \log_{(1+\alpha/2)}\left(\frac{4}{\varepsilon}\right)$. Then:*

1. *Every non-Eulerian orientation $\overrightarrow{G}$ of G contains a spring-drain path of length at most*
$$\ell_k \stackrel{\text{def}}{=} 2 \cdot \log_{(1+\alpha/2)}^{(k)} m + 2 \cdot \log_{(1+\alpha/2)}\left(\frac{4}{\varepsilon}\right). \tag{8.4}$$

2. *Every orientation $\overrightarrow{G}$ of G that is ε-far from being Eulerian is p_k-far from being Eulerian for*
$$p_k \stackrel{\text{def}}{=} \frac{\varepsilon m}{\ell_k} = \frac{\varepsilon m}{2 \cdot \log_{(1+\alpha/2)}^{(k)} m + 2 \cdot \log_{(1+\alpha/2)}\left(\frac{4}{\varepsilon}\right)}$$

Proof. We prove the lemma by induction on k. In every inductive step, we use the known bounds of ℓ_k and p_k to devise ℓ_{k+1} and p_{k+1} in an iterative manner. We start by proving the lemma for the base case, $k = 1$.

To prove Item 1 of the lemma for $k = 1$, let $\overrightarrow{G}$ be a non-Eulerian orientation of $\overrightarrow{G}$. Consider a BFS traversal of $\overrightarrow{G}$ starting from the set S of springs. For every $i \geq 0$, let L_i be the ith level of the traversal, where $L_0 = S$, and let $U_{<i} \stackrel{\text{def}}{=} \bigcup_{0 \leq j < i} L_j$ and $U_{\geq i} \stackrel{\text{def}}{=} \bigcup_{j \geq i} L_j$. For every $i > 0$, let f_i be the number of directed edges going from L_{i-1} to L_i. Let L_ℓ be the first level that contains a drain. By the expander property of G, for every $i > 0$ while $|E(U_{<i})| \leq m/2$ we have $|\partial(U_{<i})| \geq \alpha |E(U_{<i})|$. Note that for every $i \leq \ell$, the set $U_{<i}$ contains no drains, and all the directed edges that exit it are from L_{i-1} to L_i. Hence $f_i > \frac{1}{2}|\partial(U_{<i})|$. We thus obtain that

$$f_i > \frac{\alpha}{2} |E(U_{<i})|$$

for every $0 < i \leq \ell$ while $|E(U_{<i})| \leq m/2$, and therefore

$$|E(U_{<i+1})| > \left(1 + \frac{\alpha}{2}\right) |E(U_{<i})|.$$

By induction, we obtain that

$$|E(U_{<i})| > \left(1 + \frac{\alpha}{2}\right)^{i-1} f_1 \geq \left(1 + \frac{\alpha}{2}\right)^{i-1} \tag{8.5}$$

for every $0 < i \leq \ell$ for which $|E(U_{<i})| \leq m/2$. Now, if for every $0 < i \leq \ell$ we have $|E(U_{<i})| \leq m/2$, then clearly $|E(U_{<\ell})| > \left(1 + \frac{\alpha}{2}\right)^{\ell-1}$, and hence $\ell - 1 < \log_{(1+\alpha/2)} |E(U_{<\ell})| \leq \log_{(1+\alpha/2)} m$ and $\ell < \log_{(1+\alpha/2)} m$.

Otherwise, let $r > 0$ be the minimal index for which $|E(U_{<r})| > m/2$. Then, for every $r \leq i \leq \ell$ we have $|E(U_{\geq i})| < m/2$, and therefore $|\partial(U_{\geq i})| \geq \alpha |E(U_{\geq i})|$. Note that for every $i \geq r$, the set $U_{\geq i}$ contains no springs, and all the directed edges that enter it go from L_{i-1} to L_i. Therefore, $f_i > \frac{1}{2}|\partial(U_{\geq i})|$. We obtain that for every $r \leq i \leq \ell$

$$f_i > \frac{\alpha}{2} |E(U_{\geq i})|,$$

and thus

$$|E(U_{\geq i-1})| > \left(1 + \frac{\alpha}{2}\right) |E(U_{\geq i})|.$$

By induction, we have

$$|E(U_{\geq i-1})| > \left(1 + \frac{\alpha}{2}\right)^{\ell-i+1} |E(U_{\geq \ell})| \geq \left(1 + \frac{\alpha}{2}\right)^{\ell-i+1} \tag{8.6}$$

for every $r \leq i \leq \ell$.

From (8.5) and (8.6) we obtain that both $r - 1 < \log_{(1+\alpha/2)} m$ and $\ell - r + 1 < \log_{(1+\alpha/2)} m$, and therefore $\ell < 2 \cdot \log_{(1+\alpha/2)} m$. Hence, every non-Eulerian orientation of G contains a spring-drain path of length at most $\ell_1 = 2 \cdot \log_{(1+\alpha/2)} m + 2 \cdot \log_{(1+\alpha/2)} \left(\frac{4}{\varepsilon}\right)$.

To prove Item 2 of the lemma for $k = 1$, let $\overrightarrow{G}$ be an orientation of G that is ε-far from being Eulerian. While $\overrightarrow{G}$ is not Eulerian, choose a shortest spring-drain path in $\overrightarrow{G}$ and invert all its edges. By Item 1, every chosen spring-drain path is of length at most ℓ_1. Let $\overrightarrow{H}$ be the union of the spring-drain paths inverted. Clearly, $\overrightarrow{H}$ is a correction subgraph of $\overrightarrow{G}$. As $\overrightarrow{G}$ is ε-far from being Eulerian, $\overrightarrow{H}$ contains at least εm edges, and thus it is necessarily a union of at least $p_1 = \frac{\varepsilon m}{\ell_1}$ disjoint spring-drain paths. By Lemma 7.2.5, every correction subgraph of $\overrightarrow{G}$ contains the same number of disjoint spring-drain paths, which completes the proof of the base case.

Suppose now that the lemma holds for some natural $k \geq 1$ and assume that $\log^{(k)}_{(1+\alpha/2)} m \geq \log_{(1+\alpha/2)} \left(\frac{4}{\varepsilon}\right)$. The proof of both items of the lemma for $k+1$ is very similar to that of the base case. However, in Inequality (8.5) we know that $f_1 \geq p_k$, and in Inequality (8.6) we know that $|E(U_{\geq \ell})| \geq p_k$. Hence, every non-Eulerian orientation $\overrightarrow{G}$ of G contains a spring-drain path of length at most

$$\ell_{k+1} \leq 2 \cdot \log_{(1+\alpha/2)} \left(\frac{m}{p_k}\right) = 2 \cdot \log_{(1+\alpha/2)} \left(\frac{\ell_k}{\varepsilon}\right)$$

$$\leq 2 \cdot \log_{(1+\alpha/2)} \left(2 \cdot \log^{(k)}_{(1+\alpha/2)} m + 2 \cdot \log_{(1+\alpha/2)} \left(\frac{4}{\varepsilon}\right)\right) + 2 \cdot \log_{(1+\alpha/2)} \left(\frac{1}{\varepsilon}\right). \quad (8.7)$$

Since $\log^{(k)}_{(1+\alpha/2)} m \geq \log_{(1+\alpha/2)} \left(\frac{4}{\varepsilon}\right)$ we have

$$\ell_{k+1} \leq 2 \cdot \log_{(1+\alpha/2)} \left(4 \cdot \log^{(k)}_{(1+\alpha/2)} m\right) + 2 \cdot \log_{(1+\alpha/2)} \left(\frac{1}{\varepsilon}\right) = 2 \cdot \log^{(k+1)}_{(1+\alpha/2)} m + 2 \cdot \log_{(1+\alpha/2)} \left(\frac{4}{\varepsilon}\right),$$

which proves Item 1. The proof of Item 2 is the same as for the base case. □

Lemma 8.3.2. *Let G be an Eulerian α-expander. Let $\overrightarrow{G}$ be an orientation of G that is ε-far from being Eulerian. Then $\overrightarrow{G}$ is p-far from being Eulerian for*

$$p = \Omega\left(\frac{\alpha \varepsilon m}{\log(\frac{1}{\varepsilon})}\right).$$

Proof. Let k be the minimum natural number such that $\log^{(k)}_{(1+\alpha/2)} m < \log_{(1+\alpha/2)}\left(\frac{4}{\varepsilon}\right)$. Then, using the same arguments as we did in the proof of Lemma 8.3.2 for smaller k's, we obtain that every non-Eulerian orientation of G contains a spring-drain path of length at most ℓ_{k+1}, where ℓ_{k+1} satisfies Inequality (8.7). However, since $\log^{(k)}_{(1+\alpha/2)} m < \log_{(1+\alpha/2)}\left(\frac{4}{\varepsilon}\right)$, we now have

$$\ell_{k+1} < 2\cdot\log_{(1+\alpha/2)}\left(4\cdot\log_{(1+\alpha/2)}\left(\frac{4}{\varepsilon}\right)\right)+2\cdot\log_{(1+\alpha/2)}\left(\frac{1}{\varepsilon}\right) = O\left(\log_{(1+\alpha/2)}\left(\frac{1}{\varepsilon}\right)\right).$$

Similarly to our proof of Item 2 in Lemma 8.3.2, we obtain that every orientation of G that is ε-far from being Eulerian is p-far from being Eulerian for

$$p = \frac{\varepsilon m}{\ell_{k+1}} = \Omega\left(\frac{\varepsilon m}{\log_{(1+\alpha/2)}\left(\frac{1}{\varepsilon}\right)}\right) = \Omega\left(\frac{\alpha\varepsilon m}{\log(\frac{1}{\varepsilon})}\right).$$

□

Substituting the lower bound for p of Lemma 8.3.2 in Lemmas 8.1.2, 8.1.4 and 8.1.6, we obtain the following theorem.

Theorem 8.3.3. *Let G be an α-expander (for some $\alpha > 0$) with m edges and maximum degree Δ.*

1. *SIMPLE-2$\left(\overrightarrow{G}, \Omega\left(\frac{\alpha\varepsilon m}{\log(1/\varepsilon)}\right)\right)$ is a 2-sided ε-test for being Eulerian with query complexity $\widetilde{O}\left(\left(\frac{\log(1/\varepsilon)}{\alpha\varepsilon}\right)^3\right)$.*
2. *GENERIC-1$\left(\overrightarrow{G}, \Omega\left(\frac{\alpha\varepsilon m}{\log(1/\varepsilon)}\right), \Delta\right)$ is a 1-sided ε-test for being Eulerian with query complexity $O\left(\frac{\Delta\log(1/\varepsilon)}{\alpha\varepsilon}\right)$.*
3. *MULTISTAGE-2$\left(\overrightarrow{G}, \Omega\left(\frac{\alpha\varepsilon m}{\log(1/\varepsilon)}\right), \Delta\right)$ is a 2-sided ε-test for being Eulerian with query complexity $\widetilde{O}\left(\frac{\sqrt{\Delta}\log(1/\varepsilon)}{\alpha\varepsilon}\right)$.*

Note that for a constant α, the query complexity of SIMPLE-2 depends only on ε (while the other tests depend also on Δ).

8.4 Testing orientations of "lame" directed expanders

In this section we discuss a variation of the expander test, which will serve us in Section 8.5 for devising tests for general graphs. Given an orientation $\overrightarrow{G}$ of G, we now test a subgraph $\overrightarrow{G}[U]$ of $\overrightarrow{G}$, induced by a subset $U \subseteq V$. We refer to the edges in $E(U)$ as the *internal edges* of $\overrightarrow{G}[U]$, and denote $m_U \stackrel{\text{def}}{=} |E(U)|$. We say that $\overrightarrow{G}[U]$ is *Eulerian* if and only if all the vertices in U are balanced in $\overrightarrow{G}$. We say that $\overrightarrow{G}[U]$ is β-*Eulerian* if and only if all the β-small vertices in U are balanced in $\overrightarrow{G}$. Note that these definitions rely also on the edges in ∂U, which we will henceforth call *external edges*. We assume that the orientations of all the external edges are known, and furthermore, we use a distance function that does not allow inverting external edges. Namely, we will say that $\overrightarrow{G}[U]$ is ε-*close* to being Eulerian if and only if it has a correction subgraph of size at most εm_U which includes only internal edges. Otherwise, we say that $\overrightarrow{G}[U]$ is ε-*far* from being Eulerian. Similarly, we will say that $\overrightarrow{G}[U]$ is (ε, β)-*amendable* if and only if it has a β-correction subgraph of size at most εm_U which includes only internal edges. Otherwise, we say that $\overrightarrow{G}[U]$ is (ε, β)-*unamendable*. Note that we can view the external edges as comprising a knowledge graph (see Section 7.4). To ensure that $\overrightarrow{G}[U]$ can be made Eulerian (or β-Eulerian) by inverting internal edges only, we always assume that all the cuts in $\overrightarrow{G}$ are valid with respect to the orientation $\overrightarrow{\partial} U$ of the external edges. This implies in particular that

$$\overrightarrow{E}(U, V \setminus U) = \overrightarrow{E}(V \setminus U, U). \tag{8.8}$$

The next lemma shows that this assumption allows us to apply the same techniques as we did for the general testing problem.

Lemma 8.4.1. *If all the cuts in $\overrightarrow{G}$ are valid with respect to $\overrightarrow{\partial} U$, then:*

1. *$\overrightarrow{G}[U]$ can be made Eulerian by inverting internal edges along spring-drain paths.*
2. *$\overrightarrow{G}[U]$ can be made β-Eulerian by inverting internal edges along β-spring-drain paths.*
3. *If $\overrightarrow{G}[U]$ is β-Eulerian then it can be made Eulerian by inverting internal edges along spring-drain paths, where in each such path, both the spring and the drain are β-big.*

Proof. We first give a proof of Item 1, and later explain how to modify it so as to prove Item 2 and Item 3. Assume that there is a spring $s \in \overrightarrow{G}[U]$ with no path to any drain that is contained entirely in $\overrightarrow{G}[U]$. Let

$$X = \{u \in U \mid \text{ there is a directed path of internal edges from } s \text{ to } u\}.$$

As X contains no drains but at least one spring, more edges exit X than enter it. Furthermore, by the definition of X, all the edges that exit X are in $\overrightarrow{\partial} U$. Hence, the cut $(X, V \setminus X)$ is invalid with respect to $\overrightarrow{\partial} U$, a contradiction. Since inverting an internal spring-drain path does not change the orientation of the edges in ∂U, we may continue to invert such paths until $\overrightarrow{G}[U]$ becomes balanced.

To prove Item 2, note that any correction subgraph of internal edges, which exists by Item 1, contains a β-correction subgraph as a subgraph, and thus it is also internal in $\overrightarrow{G}[U]$. To obtain this β-correction subgraph, we modify the correction subgraph by removing paths from β-big springs to β-big drains one by one as long as they exist.

To prove Item 3 we use the same proof as for Item 1. However, here we know that all our springs and drains are β-big, since $\overrightarrow{G}[U]$ is β-Eulerian. □

We will be interested in induced subgraphs $\overrightarrow{G}[U]$ that are "lame directed expanders". Formally, given a subset $U \subseteq V$ and a parameter $\beta > 0$, we say that a cut (A, B) of U is a *β-cut of U* if

$$|E(B)| \geq |E(A)| \geq \beta.$$

Given parameters $\alpha, \beta > 0$, we say that the subgraph $\overrightarrow{G}[U]$ of G is an (α, β)-*expander* if for every β-cut (A, B) of U we have

$$|E(A, B)| - \left| |\overrightarrow{E}(V \setminus U, A)| - |\overrightarrow{E}(A, V \setminus U)| \right| \geq 2\alpha |E(A)|. \tag{8.9}$$

Note that to decide whether $\overrightarrow{G}[U]$ is an (α, β)-expander we do not need to know the orientation of its internal edges, but only that of its external edges. In particular, if the entire domain graph G is an α-expander, then $\overrightarrow{G}$ itself is always an (α, β)-expander for every $\beta > 0$.

In the next two lemmas we give lower bounds for the numbers of internal spring-drain paths and β'-spring-drain paths (for some β') in an (α, β)-expander. Using

our (p, β)-tests from Section 8.1 with these bounds, we will later obtain ε-tests for (α, β)-expanders. In the following, $m_U = |E(U)|$ and Δ_U is the maximum degree of a vertex in U (where the degree of a vertex $u \in U$ is the total number of edges incident with u, including external edges).

Lemma 8.4.2. *Let $\overrightarrow{G}[U]$ be an (α, β)-expander for some $0 < \alpha < 1$ and $0 \leq \beta$ and let $m_U \stackrel{\text{def}}{=} |E(U)|$. Suppose that all the cuts in G are valid with respect to $\overrightarrow{\partial} U$. Suppose that $\overrightarrow{G}[U]$ is (ε', β')-unamendable for some $0 < \varepsilon' < 1$ and $0 < \beta' \leq \Delta_U$. Then $\overrightarrow{G}[U]$ is p-far from being β'-Eulerian for $p(U, \alpha, \beta, \varepsilon', \beta') = \Omega\left(\frac{\varepsilon' m_U}{\log_{(1+\alpha)} m_U + \beta}\right) = \Omega\left(\frac{\varepsilon' m_U}{\log m_U / \alpha + \beta}\right)$.*

Proof. We show that in any orientation $\overrightarrow{G}$ such that $\overrightarrow{G}[U]$ is not β'-Eulerian, there exists a spring-drain path inside $\overrightarrow{G}[U]$ of length $O(\log_{(1+\alpha)} m_U + \beta)$. Note that an (α, β)-expander $\overrightarrow{G}[U]$ remains an (α, β)-expander after we invert some of its internal edges. Thus, if $\overrightarrow{G}[U]$ is (ε', β')-unamendable, then we have to invert internal edges along $\Omega\left(\frac{\varepsilon' m_U}{\log_{(1+\alpha)} m + \beta}\right)$ β'-spring-drain paths (as inverting spring-drains paths in which both the spring and the drain are β'-big is irrelevant for being β'-Eulerian). Thereof the lemma will follow.

Assume that $\overrightarrow{G}[U]$ is not β'-Eulerian, and let ℓ be the minimum length of a spring-drain path of internal edges in $\overrightarrow{G}[U]$. Such a path exists by Lemma 8.4.1. If $\ell \leq \beta$ then the claim is obviously true. We thus assume that $\ell > \beta$. Let S_U be the set of springs in U. Consider a BFS traversal of $\overrightarrow{G}[U]$ starting from $L_0 \stackrel{\text{def}}{=} S_U$. For every $i > 0$, set by induction

$$L_i \stackrel{\text{def}}{=} \{v \in U \mid \text{there exists } u \in L_{i-1} \text{ s.t. } (u, v) \in \overrightarrow{E}\}.$$

In addition, let $A_i \stackrel{\text{def}}{=} \bigcup_{0 \leq j < i} L_j$ and $B_i \stackrel{\text{def}}{=} \bigcup_{j \geq i} L_j$. Consider a level i such that $\beta < i \leq \ell$ and $|E(A_i)| \leq |E(B_i)|$. Clearly, (A_i, B_i) is a β-cut, and thus, since U is an (α, β)-expander, we have

$$|E(A_i, B_i)| + |\overrightarrow{E}(V \setminus U, A_i)| - |\overrightarrow{E}(A_i, V \setminus U)| \geq 2\alpha |E(A_i)|. \tag{8.10}$$

Note that for every $i \leq \ell$, the set A_i contains springs but no drains. Hence, more edges exit A_i then enter it:

$$|\overrightarrow{E}(A_i, B_i)| - |\overrightarrow{E}(B_i, A_i)| - |\overrightarrow{E}(V \setminus U, A_i)| + |\overrightarrow{E}(A_i, V \setminus U)| > 0,$$

and thus

$$|\overrightarrow{E}(A_i, B_i)| > \frac{1}{2}\left(|\overrightarrow{E}(A_i, B_i)| + |\overrightarrow{E}(B_i, A_i)| + |\overrightarrow{E}(V \setminus U, A_i)| - |\overrightarrow{E}(A_i, V \setminus U)|\right).$$

Substituting the above in Inequality (8.10) we have

$$|\overrightarrow{E}(A_i, B_i)| > \alpha|E(A_i)|$$

for every i such that $\beta < i \leq \ell$ and $|E(A_i)| \leq |E(B_i)|$. Recall that all the directed edges that exit A_i enter A_{i+1}. We thus have

$$|E(A_{i+1})| > (1+\alpha)|E(A_i)|,$$

and by induction,

$$|E(A_i)| > (1+\alpha)^{i-\beta}|E(A_\beta)| \geq (1+\alpha)^{i-\beta}\beta.$$

Therefore,

$$i - \beta \leq \log_{(1+\alpha)}\left(\frac{|E(A_i)|}{\beta}\right) < \log_{(1+\alpha)} m_U \tag{8.11}$$

for every i such that $\beta < i \leq \ell$ and $|E(A_i)| \leq |E(B_i)|$. Now, if for every $\beta < i \leq \ell$ we have $|E(A_i)| \leq |E(B_i)|$, then, from Inequality (8.11) we have $\ell - \beta < \log_{(1+\alpha)} m_U$ and thus $\ell = O(\log_{(1+\alpha)} m_U + \beta)$.

Otherwise, let $k > 0$ be the minimal index for which $|E(A_i)| > |E(B_i)|$. From Equation (8.11) we have

$$k - 1 - \beta < \log_{(1+\alpha)} m_U. \tag{8.12}$$

For every $k \leq i \leq \ell$ while $|E(B_i)| \geq \beta$, (B_i, A_i) is a β-cut, and thus from Inequality (8.9) we have

$$|E(A_i, B_i)| - |\overrightarrow{E}(V \setminus U, B_i)| + |\overrightarrow{E}(B_i, V \setminus U)| \geq 2\alpha|E(B_i)|. \tag{8.13}$$

Note that the set B_i contains drains but no springs. Hence, more edges enter B_i then exit it:

$$|\overrightarrow{E}(A_i, B_i)| - |\overrightarrow{E}(B_i, A_i)| + |\overrightarrow{E}(V \setminus U, B_i)| - |\overrightarrow{E}(B_i, V \setminus U)| > 0,$$

and thus

$$|\overrightarrow{E}(A_i, B_i)| > \frac{1}{2}\left(|\overrightarrow{E}(A_i, B_i)| + |\overrightarrow{E}(B_i, A_i)| - |\overrightarrow{E}(V \setminus U, B_i)| + |\overrightarrow{E}(B_i, V \setminus U)|\right).$$

Substituting the above in Inequality (8.13) we have

$$|\overrightarrow{E}(A_i, B_i)| > \alpha|E(B_i)|$$

for every i such that $k \leq i \leq \ell$ and $|E(A_i)| > |E(B_i)| \geq \beta$. Since all the edges in $\overrightarrow{E}(A_i, B_i)$ enter L_i, we have

$$|E(B_i)| > (1+\alpha)|E(B_{i+1})|,$$

and by induction

$$|E(B_i)| > (1+\alpha)^{j-i}|E(B_j)|$$

for every i, j such that $k \leq i \leq j \leq \ell$ and $|E(A_j)| > |E(B_j)| \geq \beta$. Hence,

$$j - i \leq \log_{(1+\alpha)}\left(\frac{|E(B_i)|}{|E(B_j)|}\right) < \log_{(1+\alpha)} m_U \tag{8.14}$$

for every i, j such that $k \leq i \leq j \leq \ell$ and $|E(A_j)| > |E(B_j)| \geq \beta$. Now, if $|E(A_\ell)| > |E(B_\ell)| \geq \beta$ then, taking $i = k$ and $j = \ell$ we have

$$\ell - k < \log_{(1+\alpha)} m_U.$$

Combined with Inequality (8.12) we obtain

$$\ell < 2\log_{(1+\alpha)} m_U + \beta + 1 = O(\log_{(1+\alpha)} m_U + \beta).$$

Otherwise, let r be the minimum index such that $|E(B_i)| < \beta$. Then, for $i = k$ and $j = r - 1$, Inequality (8.14) yields

$$r - 1 - k < \log_{(1+\alpha)} m_U. \tag{8.15}$$

Since $|E(B_r)| < \beta$, there are less then β levels between r and ℓ and thus $\ell < r + \beta$. Hence, with Inequalities (8.12) and (8.15) we achieve

$$\ell < 2\log_{(1+\alpha)} m_U + 2\beta + 2 = O(\log_{(1+\alpha)} m_U + \beta). \qquad \square$$

Lemma 8.4.3. *Let $\overrightarrow{G}[U]$ be an (α,β)-expander for some $0<\alpha<1$ and $0\leq\beta\leq\frac{\Delta_U}{2}$ and let $m_U \stackrel{\text{def}}{=} |E(U)|$. Suppose that all the cuts in G are valid with respect to $\overrightarrow{\partial}U$. Suppose further that for some $\varepsilon>0$, $\overrightarrow{G}[U]$ is ε-far from being Eulerian, but still $(\frac{\varepsilon}{2},2\beta)$-amendable. Then $\overrightarrow{G}[U]$ is p'-far from being Eulerian for $p'(U,\alpha,\beta,\varepsilon) = \Omega\left(\frac{\varepsilon m_U}{\log_{(1+\alpha)} m_U}\right) = \Omega\left(\frac{\alpha\varepsilon m_U}{\log m_U}\right)$.*

Proof. As $\overrightarrow{G}[U]$ is $(\frac{\varepsilon}{2},2\beta)$-amendable, there exists a 2β-correction subgraph $\overrightarrow{H}$ of $\overrightarrow{G}[U]$ of size at most $\varepsilon m_U/2$. Consider $\overrightarrow{G}_1[U] \stackrel{\text{def}}{=} \overrightarrow{G}[U]_{\overleftarrow{H}}$, that is, the digraph obtained from $\overrightarrow{G}[U]$ by inverting all the edges in the 2β-correction subgraph $\overrightarrow{H}$. Then $\overrightarrow{G}_1[U]$ is 2β-balanced. However, since $\overrightarrow{G}[U]$ is ε-far from being balanced, at least $\varepsilon m_U/2$ more edges must be inverted in order to make it Eulerian. By Item 3 of Lemma 8.4.1, there exists a correction subgraph for $\overrightarrow{G}[U]$ which is a union of internal paths from 2β-big springs to 2β-big drains.

To complete the proof, we show that while $\overrightarrow{G}_1[U]$ is not Eulerian, it contains an internal spring-drain path of length $\ell = O(\log_{(1+\alpha)} m_U) = O(\log m_U/\alpha)$. This is done similarly to the proof in Lemma 8.4.2 which shows the existence of a short β'-spring-drain path. However, note that now the set $E(A_2)$ includes all the edges outgoing from at least one 2β-big spring, and thus $E(A_i,B_i)$ is a β-cut for every $i\geq 2$ such that $|E(A_i)|\leq|A(B_i)|$. Hence, instead of Inequality (8.11) we have

$$i-2 \leq \log_{(1+\alpha)}\left(\frac{|E(A_i)|}{|E(A_2)|}\right) < \log_{(1+\alpha)} m_U \tag{8.16}$$

for every i such that $1<i\leq\ell$ and $|E(A_i)|\leq|E(B_i)|$. Furthermore, since all the drains are 2β-big, the set $E(B_{\ell-1})$ includes all the edges incoming to at least one 2β-big drain, and thus $E(B_i,A_i)$ is a β-cut for every $i\leq\ell-1$ such that $|E(A_i)|>|A(B_i)|$. Hence, putting $j=\ell-1$ in Inequality (8.14), we obtain

$$\ell-i-1 \leq \log_{(1+\alpha)}\left(\frac{|E(B_i)|}{|E(B_{\ell-1})|}\right) < \log_{(1+\alpha)} m_U \tag{8.17}$$

for every $i\leq\ell-1$ such that $|E(A_i)|>|A(B_i)|$. Let k be the minimum value of i for which $|E(A_i)|>|A(B_i)|$. Then, from Inequality (8.16) we have $k<\log_{(1+\alpha)} m_U+3$ and from Inequality (8.17) we have $\ell-k<\log_{(1+\alpha)} m_U+1$, and hence $\ell=O(\log_{(1+\alpha)} m_U)$. □

Suppose that $\overrightarrow{G}$ is ε-far from being Eulerian. Note that if $\beta\leq\frac{\Delta_U}{2}$, then either

the conditions of Lemma 8.4.2 apply for $\varepsilon' = \varepsilon/2$ and $\beta' = 2\beta$, or the conditions of Lemma 8.4.3 apply. Also, note that if $\beta > \frac{\Delta_U}{2}$ then the conditions of Lemma 8.4.2 apply for $\varepsilon' = \varepsilon$ and $\beta' = \Delta_U$. We thus obtain the two ε-tests below. Note that whenever our tests use samples of edges incident with a vertex $u \in U$, the sampling is among all the edges incident with u, and not only internal edges.

Algorithm 8.4.4. *GEN-1($\overrightarrow{G}[U]$, α, β, ε)*

1. *If $\beta \leq \frac{\Delta_U}{2}$ then run GENERIC-1($\overrightarrow{G}[U]$, $p(U, \alpha, \beta, \varepsilon/2, 2\beta)$, Δ_U), and otherwise run GENERIC-1($\overrightarrow{G}[U]$, $p(U, \alpha, \beta, \varepsilon, \Delta_U)$, Δ_U). In both cases, $p(U, \alpha, \beta, \varepsilon', \beta')$ is the lower bound given in Lemma 8.4.2.*
2. *If $\beta \leq \frac{\Delta_U}{2}$ then run GENERIC-1($\overrightarrow{G}[U]$, $p'(U, \alpha, \beta)$, Δ_U), where $p'(U, \alpha, \beta)$ is the lower bound given in Lemma 8.4.3.*
3. *Reject if at least one of the tests has rejected, and accept otherwise.*

Algorithm 8.4.5. *MULTI-2($\overrightarrow{G}[U]$, α, β, ε)*

1. *If $\beta \leq \frac{\Delta_U}{2}$ then run MULTISTAGE-2($\overrightarrow{G}[U]$, $p(U, \alpha, \beta, \varepsilon/2, 2\beta)$, Δ_U), and otherwise run MULTISTAGE-2($\overrightarrow{G}[U]$, $p(U, \alpha, \beta, \varepsilon, \Delta_U)$, Δ_U). In both cases, $p(U, \alpha, \beta, \varepsilon', \beta')$ is the lower bound given in Lemma 8.4.2.*
2. *If $\beta \leq \frac{\Delta_U}{2}$ then run MULTISTAGE-2($\overrightarrow{G}[U]$, $p'(U, \alpha, \beta)$, Δ_U), where $p'(U, \alpha, \beta)$ is the lower bound given in Lemma 8.4.3.*
3. *Reject if at least one of the tests has rejected, and accept otherwise.*

Combining Lemma 8.4.2 and Lemma 8.4.3 with Lemma 8.1.4 and Lemma 8.1.6, we obtain the following lemmas, which will be used in Section 8.5.

Lemma 8.4.6. *GEN-1($\overrightarrow{G}[U], \alpha, \beta, \varepsilon$) is a 1-sided ε-test for an (α, β)-expander subgraph $\overrightarrow{G}[U]$, assuming that the external edges of U are known and do not induce an invalid cut. The query complexity of the test is $O\left(\frac{\Delta_U \log m_U}{\varepsilon\alpha} + \frac{\beta \cdot \min\{\beta, \Delta_U\}}{\varepsilon}\right)$, where $m_U = |E(U)|$ and $\Delta_U = \max\{\deg(u) \mid u \in U\}$.*

Lemma 8.4.7. *MULTI-2($\overrightarrow{G}[U], \alpha, \beta, \varepsilon$) is a 2-sided ε-test for an (α, β)-expander subgraph $\overrightarrow{G}[U]$, assuming that the external edges of U are known and do not induce an invalid cut. The query complexity of the test is $\widetilde{O}\left(\frac{\sqrt{\Delta_U} \log m_U}{\varepsilon\alpha} + \frac{\beta \cdot \sqrt{\min\{\beta, \Delta_U\}}}{\varepsilon}\right)$, where $m_U = |E(U)|$ and $\Delta_U = \max\{\deg(u) \mid u \in U\}$.*

8.5 General tests based on chopping

In this section we use our results from Section 8.4 to provide a 1-sided test and a 2-sided test as follows. Given an orientation $\overrightarrow{G}$ of an Eulerian graph G, we show how to decompose $\overrightarrow{G}$ into a collection of (α, β)-expanders with a relatively small number of edges that are outside the (α, β)-expanders, called henceforth *external edges*. We will find this "chopping" adaptively while querying external edges only. If we do not find a witness showing that $\overrightarrow{G}$ is not Eulerian already during the chopping procedure, then we sample a few (α, β)-expanders and test them using GEN-1 (Algorithm 8.4.4) or using MULTI-2 (Algorithm 8.4.5), obtaining a 1-sided test or a 2-sided test respectively.

Lemma 8.5.1 (The chopping lemma)**.** *Given an orientation $\overrightarrow{G}$ as input and parameters $\alpha, \beta > 0$, we can either find a witness showing that $\overrightarrow{G}$ is not Eulerian, or find non-empty induced subgraphs $\overrightarrow{G}_i = (V_i, \overrightarrow{E}_i = \overrightarrow{E}(V_i))$ of $\overrightarrow{G}$ (where $i = 1, \ldots, k$ for some k), which we call (α, β)-*components *(or simply* components*), that satisfy the following:*

1. *The vertex sets $V_1, \ldots, V_k$ of the components are mutually disjoint.*
2. *$|\overrightarrow{E}_i| \geq \beta$ for $i = 1, \ldots k$.*
3. *All the components $\overrightarrow{G}_i$ are (α, β)-expanders.*
4. *The total number of external edges satisfies*

$$|\overrightarrow{E} \setminus \bigcup_{i=1,\ldots,k} \overrightarrow{E}(V_i)| = O(\alpha m^2 \log m/\beta).$$

During the chopping procedure, we query only external edges, i.e., edges that are not in any component G_i. The query complexity is of the same order also if we find a witness that $\overrightarrow{G}$ is not Eulerian.

Proof. The chopping procedure proceeds as follows. At first, we define $\overrightarrow{G} = \overrightarrow{G}[V]$ as our single component. Then, in each step, we decompose a component $\overrightarrow{G}[U]$ into two separate components $\overrightarrow{G}[A]$ and $\overrightarrow{G}[B]$, if (A, B) is a β-cut of U and

$$|E(A,B)| - \left||\overrightarrow{E}(V \setminus U, A)| - |\overrightarrow{E}(A, V \setminus U)|\right| < 2\alpha|E(A)|. \tag{8.18}$$

When decomposing, we query the edges of the cut (A, B) and mark them as external edges. Note that we need not query any additional edges to decide on cutting a component, as all the required information is given by the domain graph G and the orientation of the external edges that were queried in previous steps. After each stage, we check whether the orientations of the edges queried so far invalidate any of the cuts in the graph (see Section 7.4), in which case we conclude that $\overrightarrow{G}$ is not Eulerian and return the invalid cut.

The procedure terminates once there is no cut of any component that satisfies the chopping conditions. The components are clearly disjoint throughout the procedure. Since we only chopped components across β-cuts, every final component contains at least β edges. Moreover, note that a component is always chopped by the procedure unless all its β-cuts satisfy Inequality (8.9). Hence, if the algorithm terminates without finding a witness that $\overrightarrow{G}$ is not Eulerian, then every $\overrightarrow{G}_i$ is an (α, β)-expander.

It remains to prove the upper bound for the number of external edges and the query complexity of the chopping procedure. Consider a component U and a β-cut (A, B) of U that was queried in some step of the lemma. Suppose that the cut $(A, V \setminus A)$ is valid. Then

$$|\overrightarrow{E}(A, B)| - |\overrightarrow{E}(B, A)| + |\overrightarrow{E}(A, V \setminus U)| - |\overrightarrow{E}(V \setminus U, A)| = 0. \tag{8.19}$$

Combining this with the chopping condition (8.18), and considering two cases depending on whether $|\overrightarrow{E}(A, V \setminus U)| - |\overrightarrow{E}(V \setminus U, A)|$ is positive or negative, we obtain that

$$\min\left\{|\overrightarrow{E}(A, B)|, |\overrightarrow{E}(B, A)|\right\} < \alpha|E(A)|. \tag{8.20}$$

Hence, if Inequality (8.20) does *not* hold, then, in any case, after querying the edges in (A, B) we discover an invalid cut in the graph (which could be (A, B) or another cut). We next compute the query complexity in the case where our knowledge graph contains no invalid cuts throughout the procedure, and later show how to modify our analysis for the case where an invalid cut is detected after querying the last cut.

For every β-cut (A, B) that we use for partitioning, we refer to the edges in the minimal cut among $\overrightarrow{E}(A, B)$ and $\overrightarrow{E}(B, A)$ as *rare* edges, and to the edges in the other direction as *common* edges. Let us first compute the cost of querying the rare edges only. For every partition of a β-cut (A, B), we "charge" a cost of α on every edge in $E(A)$. From Inequality (8.20), the sum of charges is larger than the number

of rare edges queried. Since by our definition $|E(A)| \leq |E(B)|$, every edge can belong to "Side" A of a partitioned β-cut (A, B) at most $\log m$ times throughout the entire procedure. Hence, the sum of charges is at most $\alpha \log m$ for every edge in the graph and at most $\alpha m \log m$ in total. We complete the proof by showing that the ratio between the number of external edges and the number of rare external edges is $O(m/\beta)$.

Consider the component multigraph $\overrightarrow{G}_{comp}$, whose vertices are the components G_i and whose edges are the external edges of $\overrightarrow{G}$. By our assumption that the knowledge graph contains no invalid cuts, $\overrightarrow{G}_{comp}$ is Eulerian (as a directed multigraph), and hence it is an edge-disjoint union of simple directed cycles. However, $\overrightarrow{G}_{comp}$ does not contain a directed cycle of common edges. This can proved by induction as follows. When starting the chopping procedure there is only one component and no external edges. Then at every step we partition one component into two sets, and add the common edges, if any, from one of the sets to the other. One can see that this cannot create directed cycles of common edges. Therefore, $\overrightarrow{G}_{comp}$ is an edge-disjoint union of simple directed cycles, where each cycle contains at least one rare edge. As the number of vertices in $\overrightarrow{G}_{comp}$ is at most m/β, we conclude that the number of external edges in $\overrightarrow{G}$ is at most m/β times the number of rare edges. From the discussion above, it follows that the number of external edges in $\overrightarrow{G}$ is $O(\alpha m^2 \log m/\beta)$.

Regarding the case where querying the edges of a β-cut (A, B) reveals a violation of a cut in the graph, recall that as long as the knowledge graph does not induce an invalid cut, there exists an Eulerian orientation extending the knowledge graph (see Lemma 7.4.2). Thus, the edges of (A, B) have a "good" orientation that does not violate any cut in the graph. Clearly, the total query complexity after we query the edges of (A, B) and terminate is no higher than in the case where we would have queried the edges of (A, B) and discovered the good orientation. $\square$

We are now ready to present our 1-sided test. In the following, let $\alpha, \beta > 0$ be parameters.

Algorithm 8.5.2. *CHOP-1($\overrightarrow{G}, \varepsilon, \alpha, \beta$):*

1. *Use Lemma 8.5.1 (the chopping lemma) for finding (α, β)-components $\overrightarrow{G}_1, \ldots, \overrightarrow{G}_k$ and querying their external edges, or reject and terminate if an invalid cut is found in the process.*

2. *Sample* $3 \ln 3/\varepsilon$ (α, β)*-components* $\overrightarrow{G}_i$ *randomly and independently, where the probability of selecting a component* $\overrightarrow{G}_i$ *in each sample is proportional to* $m_i \stackrel{\text{def}}{=} |E(V_i)|$.

3. *Test every selected component* $\overrightarrow{G}_i$ *using GEN-1(*$\overrightarrow{G}_i, \alpha, \beta, \varepsilon/2$*) (Algorithm 8.4.4). Reject if the test rejects for at least one of the components selected.*

4. *Accept if the input was not rejected by any of the above steps.*

Lemma 8.5.3. *If* $\overrightarrow{G}$ *is Eulerian then CHOP-1 accepts* $\overrightarrow{G}$ *with probability 1.*

Proof. If $\overrightarrow{G}$ is Eulerian then $\overrightarrow{G}$ has no invalid cuts and therefore CHOP-1 does not reject in Step 1. In addition, every (α, β)-component of $\overrightarrow{G}$ is Eulerian, and since GEN-1 is 1-sided (see Lemma 8.4.6), CHOP-1 does not reject any of the tested (α, β)-components. □

Lemma 8.5.4. *If the external edges induce an invalid cut, or if more than an* $\varepsilon/2$*-fraction of the internal edges are in* (α, β)*-components that are* $\varepsilon/2$*-far from being Eulerian, then CHOP-1 rejects with probability at least* $2/3$.

Proof. If the external edges induce an invalid cut then the algorithm rejects in Step 1 while trying to perform the chopping. Otherwise, every (α, β)-component sampled in Step 2 is $\varepsilon/2$-far from being Eulerian with probability at least $\varepsilon/2$. By Lemma 8.4.6, for every (α, β)-component that is $\varepsilon/2$-far from being Eulerian, the rejection probability is at least $2/3$. Thus, the probability of rejecting one sampled component is at least $\varepsilon/3$. Since the components are selected independently, the probability of accepting all the components is at most $(1-\varepsilon/3)^{3 \ln 3/\varepsilon} < e^{-\ln 3} = 1/3$. □

Lemma 8.5.5. *If the external edges do not induce an invalid cut and at most an* $\varepsilon/2$*-fraction of the internal edges are in* (α, β)*-components that are* $\varepsilon/2$*-far from being Eulerian, then* $\overrightarrow{G}$ *is* ε*-close to being Eulerian.*

Proof. By Item 1 of Lemma 8.4.1, every (α, β)-component $\overrightarrow{G}_i$ in $\overrightarrow{G}$ can be made Eulerian by inverting internal edges of $\overrightarrow{G}_i$. We thus orient the (α, β)-components that are $\varepsilon/2$-far from being Eulerian so as to make them Eulerian. These components consist of at most $\varepsilon m/2$ edges. In addition, we invert a minimum number of edges in each of the (α, β)-components that are $\varepsilon/2$-close to being Eulerian, so as to make them Eulerian too. This requires at most $\varepsilon m/2$ more alterations. We thus obtain an Eulerian orientation that is ε-close to $\overrightarrow{G}$. □

Theorem 8.5.6. *CHOP-1 is a 1-sided test for being Eulerian with query complexity*

$$O\left(\frac{\alpha m^2 \log m}{\beta} + \frac{\Delta \log m}{\varepsilon^2 \alpha} + \frac{\beta \cdot \min\{\beta, \Delta\}}{\varepsilon^2}\right).$$

In particular, for $\alpha = \frac{(\Delta \log m)^{1/3}}{(\varepsilon m)^{2/3}}$ *and* $\beta = \frac{(\varepsilon m \log m)^{2/3}}{\Delta^{1/3}}$*, the query complexity is*

$$O\left(\frac{(\Delta m \log m)^{2/3}}{\varepsilon^{4/3}}\right) = O\left(\left(\frac{\Delta}{\varepsilon}\right)^{4/3} (n \log n)^{2/3}\right).$$

Proof. The correctness of the test follows from Lemmas 8.5.3, 8.5.4, and 8.5.5. The query complexity follows from Lemmas 8.4.6 and 8.5.1 (the chopping lemma). □

We note that Theorem 8.5.6 provides a sub-linear algorithm for every graph of maximum degree $\Delta = o\left(\frac{\varepsilon^2 \sqrt{m}}{\log m}\right)$. For nearly regular graphs, i.e. graphs with $m = \Omega(\Delta n)$, the algorithm is sub-linear for every $\Delta = o\left(\frac{\varepsilon^4 n}{\log^2 n}\right)$.

We conclude with a similar 2-sided test which gives a sub-linear query complexity for all graphs. In the following, let $\alpha, \beta > 0$ be parameters.

Algorithm 8.5.7. *CHOP-2($\overrightarrow{G}, \varepsilon, \alpha, \beta$):*

1. *Use Lemma 8.5.1 (the chopping lemma) for finding (α, β)-components $\overrightarrow{G}_1, \ldots, \overrightarrow{G}_k$ and querying their external edges, or reject and terminate if an invalid cut is found in the process.*

2. *Sample $\frac{3}{\varepsilon}$ (α, β)-components $\overrightarrow{G}_i$ independently, where the probability of selecting a component $\overrightarrow{G}_i$ is proportional to $m_i \stackrel{\text{def}}{=} |E(V_i)|$.*

3. *Test every selected component $\overrightarrow{G}_i$ for being Eulerian $12 \ln(9/\varepsilon)$ times independently using MULTI-2($\overrightarrow{G}_i, \alpha, \beta, \varepsilon/2$) (Algorithm 8.4.5). Reject if there is a component $\overrightarrow{G}_i$ which was rejected by at least half of its tests.*

4. *Accept if the input was not rejected in a previous step.*

Lemma 8.5.8. *If $\overrightarrow{G}$ is Eulerian then CHOP-2 accepts with probability at least $2/3$.*

Proof. If $\overrightarrow{G}$ is Eulerian then $\overrightarrow{G}$ has no invalid cuts and therefore CHOP-2 does not reject in Step 1. In addition, all the (α, β)-components $\overrightarrow{G}_i$ are Eulerian. Thus,

by Lemma 8.4.7, every run of MULTI-2 on a component $\overrightarrow{G}_i$ rejects with probability at most 1/3. By standard large deviation arguments, the probability of rejecting a component $\overrightarrow{G}_i$ by at most half of its tests is at most $\varepsilon/9$. Applying the union bound for the $3/\varepsilon$ (α, β)-components sampled, the probability of rejecting $\overrightarrow{G}$ is at most 1/3. □

Lemma 8.5.9. *If the external edges induce an invalid cut, or if more than $\varepsilon/2$-fraction of the internal edges are in (α, β)-components that are $\varepsilon/2$-far from being Eulerian, then Algorithm 8.5.7 rejects with probability at least 2/3.*

Proof. If the external edges induce an invalid cut then the algorithm rejects in Step 1 while trying to perform the chopping. Otherwise, the probability of not sampling any (α, β)-component that is $\varepsilon/2$-far from being Eulerian is at most $(1 - \frac{\varepsilon}{2})^{3/\varepsilon} < \frac{1}{4}$. Suppose that we have sampled at least one (α, β)-component that is ε-far from being Eulerian. By Lemma 8.4.7, the acceptance probability of a single test of this component is at most 1/3. Using standard large deviation arguments, the probability of accepting in at most half of the tests of this component is smaller than 1/12. We conclude that the probability of accepting $\overrightarrow{G}$ in this case is at most 1/3. □

Observation 8.5.10. *Lemma 8.5.5 is true for CHOP-2 as well as for CHOP-1.* □

Theorem 8.5.11. *Algorithm 8.5.7 is a 2-sided test for being Eulerian with query complexity*

$$O\left(\frac{\alpha m^2 \log m}{\beta}\right) + \widetilde{O}\left(\frac{\sqrt{\Delta}\log m}{\varepsilon^2 \alpha} + \frac{\beta \cdot \sqrt{\min\{\beta, \Delta\}}}{\varepsilon^2}\right).$$

In particular, if $\Delta \leq (\varepsilon m)^{4/7}$, then for $\alpha = \frac{\Delta^{1/6}}{(\varepsilon m)^{2/3}}$ and $\beta = \frac{(\varepsilon m)^{2/3}}{\Delta^{1/6}}$ the query complexity is $\widetilde{O}\left(\frac{\Delta^{1/3} m^{2/3}}{\varepsilon^{4/3}}\right) = \widetilde{O}\left(\frac{m^{6/7}}{\varepsilon^{8/7}}\right)$. If $(\varepsilon m)^{4/7} < \Delta \leq m$, then for $\alpha = \frac{\Delta^{5/16}}{(\varepsilon m)^{3/4}}$ and $\beta = \Delta^{1/8}\sqrt{\varepsilon m}$ the query complexity is $\widetilde{O}\left(\frac{\Delta^{3/16} m^{3/4}}{\varepsilon^{5/4}}\right) = \widetilde{O}\left(\frac{m^{15/16}}{\varepsilon^{5/4}}\right)$.

Proof. The correctness of the test follows from Lemmas 8.5.8, 8.5.9, and 8.5.5. The query complexity follows from Lemmas 8.4.7 and 8.5.1 (The chopping lemma). □

8.6 Discussion

The procedure of our general test is surprisingly involved considering the problem statement. The question arises as to whether we can reduce the computational complexity from exponential to polynomial in m or to perform most of the calculations in a preprocessing stage. Another unresolved question is whether there exists a non-adaptive sub-linear test for all graphs.

Also, it would be interesting to consider a generalized models where each edge may have a weight affecting the distance function or a capacity in addition to a direction. The latter may describe natural network flow problems where a flow between two nodes may be of varying volume as well as direction. Such generalizations would require more evolved techniques, since our BFS-based analysis does not immediately apply to them in its current form.

Chapter 9

Lower Bounds for Bounded-Degree Graphs

9.1 A 2-sided lower bound

In this section we prove the following theorem.

Theorem 9.1.1. *For every $0 < \varepsilon \leq 1/64$, every non-adaptive (2-sided) ε-test for Eulerian orientations of bounded degree graphs must use $\Omega\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ queries. Consequently, every adaptive test requires $\Omega(\log\log m)$ queries.*

The main idea of the proof uses Yao's principle [53]. Namely, for infinitely many natural numbers ℓ, we define a graph G_ℓ with $m = 2\ell^2$ edges and two distributions over the orientations of G_ℓ. The first distribution, $\mathcal{P}_\ell$, contains only Eulerian orientations of G_ℓ, while the second distribution, $\mathcal{F}_\ell$, contains orientations that are with high probability 1/64-far from being Eulerian. We then show that any non-adaptive deterministic algorithm which makes $o\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ queries cannot distinguish between the distributions $\mathcal{P}_\ell$ and $\mathcal{F}_\ell$ with probability higher than 1/5.

All our underlying graphs G_ℓ are two dimensional *tori*, which are 4-regular graphs having a highly symmetric structure (the exact definition is given below). We exploit this symmetry to construct distributions $\mathcal{P}_\ell$ and $\mathcal{F}_\ell$ such that, for any fixed set Q of $o\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ edges, with high probability, the orientation of every pair of edges in Q has either no correlation in any of the distributions, or a correlation that is identical in both distributions.

We build the orientations in $\mathcal{P}_\ell$ and $\mathcal{F}_\ell$ from repeated "patterns" of varying sizes and show that, in order to distinguish between the distributions, a deterministic

algorithm must be approximately synchronized with the (unknown) size of these patterns.

9.1.1 Preliminaries

For $i, j \in [\ell]$ we let $i \oplus j$ denote addition modulo ℓ, that is:

$$i \oplus j = \begin{cases} i+j & , \ i+j \leq \ell \\ i+j-\ell & , \ i+j > \ell \end{cases}.$$

Given a graph $G = (V, E)$ and two vertices $u, v \in V$, we define the *distance* between u and v (or shortly $\text{dist}(u, v)$), as the the walking distance in G between u and v. Given two edges $e_1, e_2 \in E$, we define the *distance* between e_1 and e_2 (or shortly $\text{dist}(e_1, e_2)$) as the minimal distance between an endpoint of e_1 and an endpoint of e_2. For an edge $e = \{u, v\} \in E$ and a vertex $w \in V$, we define the *distance* of e from w, or shortly $\text{dist}(e, w)$, as the minimum of $\text{dist}(u, w)$ and $\text{dist}(v, w)$. We stress that even when we consider an orientation $\overrightarrow{G}$, the distances between edges and vertices are still measured on the underlying undirected graph G for the purpose of the following proofs.

Definition 9.1.2 (Torus)**.** *A torus is a two dimensional cyclic grid. Formally, an* $\ell \times \ell$ torus *is the graph* $T = (V, E)$ *on* $n = \ell^2$ *vertices* $V = \{v_{i,j} : i, j \in [\ell]\}$ *and* $m = 2\ell^2$ *edges* $E = E_H \cup E_V$*, where* $E_H = \left\{\{v_{i,j_1}, v_{i,j_2}\} : j_2 = j_1 \oplus 1\right\}$ *and* $E_V = \left\{\{v_{i_1,j}, v_{i_2,j}\} : i_2 = i_1 \oplus 1\right\}$*. We refer to* E_H *as the set of* horizontal edges *and to* E_V *as the set of* vertical edges*. Two edges* $e_1, e_2 \in E$ *are said to be* perpendicular *if one of them is horizontal and the other is vertical, and otherwise they are called* parallel*.*

Given an orientation $\overrightarrow{T}$ *of* T*, we say that a horizontal edge* $e = \{v_{i,j}, v_{i,j\oplus 1}\}$ *is* directed to the right *if* $v_{i,j}$ *is the start-point of* e*, and otherwise we say that* e *is* directed to the left*. Similarly, we say that a vertical edge* $e = \{v_{i,j}, v_{i\oplus 1,j}\}$ *is* directed upwards *if* $v_{i,j}$ *is the start-point of* e*, and otherwise we say it is* directed downwards*.*

To simplify the presentation, we assume throughout this section that ℓ is even. We now define a graph operation that will be used later in the construction of the distributions $\mathcal{P}_\ell$ and $\mathcal{F}_\ell$.

Definition 9.1.3 ((a, b)-shifting)**.** *Let* $\overrightarrow{T}$ *be an orientation of an* $\ell \times \ell$ torus T*, and let* $a, b \in [\ell]$*. We define the* (a, b)-shifting of $\overrightarrow{T}$ *to be the orientation* $\overrightarrow{T_{a,b}}$ *of* T*, which*

is a transformation of the orientation $\overrightarrow{T}$ *a units upwards and* b *units rightward. Namely, for every edge* e *of* T*, if* $e = \{v_{i,j}, v_{i',j'}\}$ *is directed from* $v_{i,j}$ *to* $v_{i',j'}$ *in* $\overrightarrow{T}$ *then* $e_{a,b} \stackrel{\text{def}}{=} \{v_{i\oplus a,j\oplus b}, v_{i'\oplus a,j'\oplus b}\}$ *is directed from* $v_{i\oplus a,j\oplus b}$ *to* $v_{i'\oplus a,j'\oplus b}$ *in* $\overrightarrow{T_{a,b}}$.

9.1.2 Defining auxiliary distributions

In this subsection, let $H = (V, E)$ be an $\ell \times \ell$ torus, where $V = \{v_{i,j} \mid i, j \in [\ell]\}$, using the same indexing as in Definition 9.1.2. We define two simple distributions, $\mathcal{R}_\ell$ and $\mathcal{C}_\ell^{(k)}$, over the orientations of H. We later use these distributions to build the final distributions, $\mathcal{F}_\ell$ and $\mathcal{P}_\ell$.

The distribution $\mathcal{R}_\ell$ is simply a random orientation of H's edges. Namely, in $\overrightarrow{H} \sim \mathcal{R}_\ell$ the orientation of every edge $e \in E$ is chosen uniformly at random, independently of the other edges.

Lemma 9.1.4. *Let* $\overrightarrow{H}$ *be an orientation of* H*, chosen according to the distribution* $\mathcal{R}_\ell$*. Then with probability* $1 - o(1)$*, there are at least* $\ell^2/4$ *unbalanced vertices in* $\overrightarrow{H}$.

Proof. Define a subset $I = \{v_{i,j} \mid i + j \text{ is even}\} \subseteq V$ of $\ell^2/2$ vertices. Observe that I is an independent set in H (i.e. it has no internal edges), and so every vertex $v_i \in I$ is balanced with probability $x_i = \binom{4}{2}\left(\frac{1}{2}\right)^4 = 3/8$ independently from all other vertices in I. By Chernoff's inequality, the probability that at least half of the vertices in I are balanced is bounded by $\exp(-\ell^2/64)$. Namely, with probability $1 - o(1)$ there are at least $\ell^2/4$ unbalanced vertices in I. □

The second distribution, $\mathcal{C}_\ell^{(k)}$, is over Eulerian orientations of H. We assume that $2k$ divides $\ell = \sqrt{m/2}$. To construct an orientation according to $\mathcal{C}_\ell^{(k)}$, we first partition the edges of H into edge-disjoint "square-shaped" $4k$-cycles as follows. For every $0 \le i < \ell$, $0 \le j < \ell/2k$, we let $v_{i,2kj\oplus i}$ be a "lower-left corner" of a cycle C. The other "corner" vertices of C are $v_{i\oplus k,2kj\oplus i}$, $v_{i\oplus k,2kj\oplus i\oplus k}$, and $v_{i,2kj\oplus i\oplus k}$. The four corner vertices are connected by two paths of k horizontal edges and two paths of k vertical edges. One can see that this indeed forms a partition of all H's edges into edge-disjoint cycles. Then, for every cycle C, we randomly and independently choose one of C's two possible Eulerian orientations. Let $\overrightarrow{H}'$ denote the orientation of H at this stage. Finally, $a, b \in [\ell]$ are chosen uniformly at random, and $\overrightarrow{H}$ is set to be the (a, b)-shifting of $\overrightarrow{H}'$.

In what follows, for a pair of edges $e_i, e_j \in E$ and an orientation $\overrightarrow{H}$ of H, we say that e_i and e_j are *independent* if either $\overrightarrow{H} \sim \mathcal{R}_\ell$, or $\overrightarrow{H} \sim \mathcal{C}_\ell^{(k)}$ and the edges e_i

and e_j reside in different $4k$-cycles C_i and C_j. A set $Q \subseteq E$ is called *independent* if all the pairs $e_1, e_2 \in Q$ are independent. Observe that if Q is independent, then the orientation of every $e \in Q$ is distributed uniformly at random, independently of the orientation of all other members of Q. Clearly, if $\overrightarrow{H}$ is distributed according to $\mathcal{R}_\ell$ then every set $Q \subseteq E$ is independent. In the following lemmas we prove that, under some conditions, the set Q is independent with high probability also if $\overrightarrow{H}$ is distributed according to $\mathcal{C}_\ell^{(k)}$.

Lemma 9.1.5. *Let $e_1, e_2 \in E$ be two perpendicular edges of H. Let $\overrightarrow{H}$ be an orientation of H distributed according to $\mathcal{C}_\ell^{(k)}$, for an integer k that divides $\ell/2$. Then the probability that e_1 and e_2 are independent is at least $1 - \frac{1}{2k}$.*

Proof. Suppose that e_1 and e_2 are not independent. Hence, they both reside in the same cycle C in the partition of H's edges into $4k$-cycles. Note that e_1 and e_2 can define a unique square-shaped $4k$-cycle in which they can both reside, and hence, they define a unique vertex in H that must be the lower-left corner of this cycle. By the definition of the partition into $4k$-cycles, it is easy to see that the fraction of vertices in H that are corner vertices is $\frac{1}{2k}$. The lemma follows since in the last stage of constructing an orientation from $\mathcal{C}_\ell^{(k)}$, the partition into $4k$-cycles is randomly shifted. □

Lemma 9.1.6. *Let k be an integer that divides $\ell/2$. Let $Q \subseteq E$ be a set of $o(\sqrt{k})$ edges such that for every pair $e_1, e_2 \in Q$, either $\mathrm{dist}(e_1, e_2) > 2k$, or e_1 and e_2 are perpendicular. Then for an orientation $\overrightarrow{H}$ of H distributed according to $\mathcal{C}_\ell^{(k)}$, the probability that Q is independent is $1 - o(1)$.*

Proof. Fix a pair $e_1, e_2 \in Q$. If $\mathrm{dist}(e_1, e_2) > 2k$ then e_1 and e_2 must reside in different $4k$-cycles, and hence they are independent. Otherwise, e_1, e_2 are perpendicular, and by Lemma 9.1.5 they are independent with probability at least $1 - \frac{1}{2k}$. The proof is now completed by applying the union bound for all $o(k)$ pairs $e_1, e_2 \in Q$. □

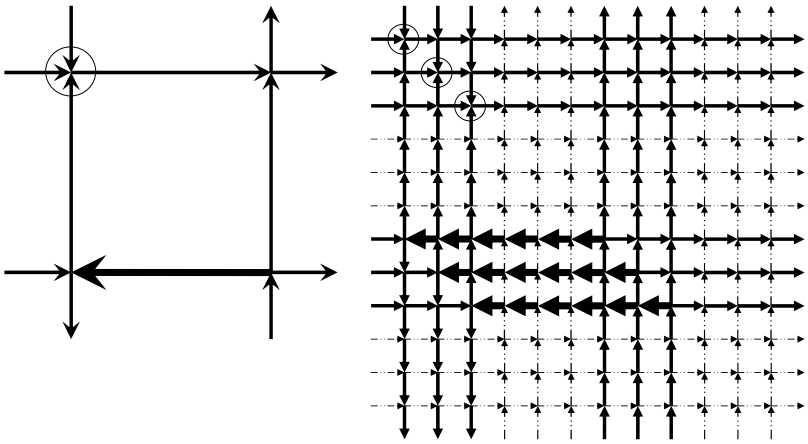
9.1.3 Defining the main distributions

In this subsection we give two distributions of torus orientations. First, we need to define the following operation.

Definition 9.1.7 (t-tiling)**.** *Let $\ell, t > 0$. Let $\overrightarrow{H} = (V(H), \overrightarrow{E}(H))$ be an $\ell \times \ell$ directed torus where $V(H) = \{v_{i,j} \mid i, j \in [\ell]\}$. Let $T = (V, E)$ be an $2t\ell \times 2t\ell$ torus where $V = \{u_{i,j} \mid i, j \in [2t\ell]\}$.*

We define the t-tiling of $\overrightarrow{H}$ *as the orientation $\overrightarrow{H}^t$ of T which is constructed as follows. First, partition T into ℓ^2 disjoint $2t \times 2t$ grids $\{G_{i,j}\}_{i,j \in [\ell]}$, where every grid $G_{i,j}$ is associated with the vertex $v_{i,j} \in V(H)$. Formally, For every $i, j \in [\ell]$ the grid $G_{i,j}$ is the induced subgraph of T whose set of vertices is $V_{i,j} \stackrel{\text{def}}{=} \{u_{i',j'} : 2t(i-1) < i' \leq 2ti,\ 2t(j-1) < j' \leq 2tj\}$. The upper left $t \times t$ grid of every $G_{i,j}$ is denoted by $R_{i,j}$ and is called the* representative grid *of the vertex $v_{i,j} \in V(H)$.*

Figure 9.1: A directed 2×2 torus $\overrightarrow{H}$ (left) and its corresponding 3-tiling, $\overrightarrow{T}$. *The vertex $v_{2,1}$ is encircled in $\overrightarrow{H}$, and its corresponding vertices $r^1_{2,1}, r^2_{2,1}$ and $r^3_{2,1}$ are encircled in $\overrightarrow{T}$. In addition, the edge $\{v_{1,2}, v_{1,1}\}$ is emphasized in $\overrightarrow{H}$, and its corresponding edges are emphasized in $\overrightarrow{T}$. Note the circular orientation of the padding edges in $\overrightarrow{T}$, marked with dashed arrows.*



The orientation $\overrightarrow{H}^t$ of T is defined as follows. For every $v_{i,j} \in V(H)$, let $r^1_{i,j}, r^2_{i,j}, \ldots, r^t_{i,j} \in V$ be the t vertices on the main diagonal of the representative grid $R_{i,j}$. For every edge $e = \{v_{i,j}, v_{i',j'}\} \in \overrightarrow{E}(H)$ directed from $v_{i,j}$ to $v_{i',j'}$ and every $h \in [t]$, we orient the edges on the shortest path from $r^h_{i,j}$ to $r^h_{i',j'}$ in a way that forms a directed path from $r^h_{i,j}$ to $r^h_{i',j'}$. For every edge $e' \in E$ that participates in this path, we call e the originating edge *of e', and use the notation $\text{org}(e') \stackrel{\text{def}}{=} e$. The edges e' of T originated in this manner are called* representative edges*, whereas*

the remaining edges are called padding edges*. Next, all the horizontal padding edges are directed to the right, and all the vertical padding edges are directed upwards (see Definition 9.1.2). See an example in Figure 9.1. For every padding edge e we define* $\mathrm{org}(e) \stackrel{\text{def}}{=} \emptyset$*, since they have no origin in* H*.*

The next lemma states that a tiling of an Eulerian torus is also Eulerian, while on the other hand, a tiling of a torus with many unbalanced vertices results with a torus that is far from being Eulerian.

Lemma 9.1.8. *Let* $\overrightarrow{H} = (V(H), \overrightarrow{E}(H))$ *be a directed* $\ell \times \ell$ *torus and let* $\overrightarrow{H}^t = (V, \overrightarrow{E})$ *be the* t*-tiling of* $\overrightarrow{H}$ *for some natural number* t*. Then,*

- *If* $\overrightarrow{H}$ *is Eulerian, then* $\overrightarrow{H}^t$ *is also Eulerian.*
- *For every* $0 < \delta < 1$*, if* $\overrightarrow{H}$ *contains* $\delta\ell^2$ *unbalanced vertices, then* $\overrightarrow{H}^t$ *is* $\frac{\delta}{16}$*-far from being Eulerian.*

Proof. The first statement of the lemma follows easily from Definition 9.1.7. Assume now that $\overrightarrow{H}$ has $\delta\ell^2$ unbalanced (spring or drain) vertices. According to the definition of a t-tiling, for every unbalanced vertex $v_{i,j} \in V(H)$ we have exactly t unbalanced vertices $r^1_{i,j}, r^2_{i,j}, \ldots, r^t_{i,j}$ on the main diagonal of $v_{i,j}$'s representative grid $R_{i,j}$ in $\overrightarrow{H}^t$, so the number of unbalanced vertices in $\overrightarrow{H}^t$ is $\delta\ell^2 t$. In addition, whenever $v_{i,j}$ is a spring (respectively drain) vertex in $\overrightarrow{H}$, the vertices $r^1_{i,j}, r^2_{i,j}, \ldots, r^t_{i,j}$ are also springs (respectively drains) in $\overrightarrow{H}^t$, so every pair of spring-drain vertices must reside in different grids $R_{i,j}$ and $R_{i',j'}$. This implies that (due to the orientation of the padding edges) the distance from any spring vertex to any drain vertex in $\overrightarrow{H}^t$ is at least t. Consequently, every correction path in $\overrightarrow{H}^t$ must be of length at least t. Since every correction path in $\overrightarrow{H}^t$ can balance at most two unbalanced vertices, and since the length of every such path is at least t, we conclude that $\overrightarrow{H}^t$ is $\frac{t\delta\ell^2 t/2}{|E(\overrightarrow{H}^t)|} = \frac{\delta\ell^2 t^2/2}{8\ell^2 t^2} = \frac{\delta}{16}$-far from being Eulerian. □

Lemma 9.1.9. *Let* $\overrightarrow{H}^t$ *be a* t*-tiling of a randomly oriented* $\ell \times \ell$ *torus* $\overrightarrow{H} \sim \mathcal{R}_\ell$*. Then with probability* $1 - o(1)$*,* $\overrightarrow{H}^t$ *is* $1/64$*-far from being Eulerian.*

Proof. Follows by combining Lemma 9.1.8 (with $\delta = 1/4$) and Lemma 9.1.4. □

We are now ready to define the distributions $\mathcal{P}_\ell$ and $\mathcal{F}_\ell$ over the orientations of an $\ell \times \ell$ torus $T = (V, E)$. To avoid divisibility concerns, we assume that $\ell = 2^k$ and $k = 2^b$ for some natural number $b > 1$. It is easy to verify that the same proof works also for general values of ℓ and k by using rounding as appropriate.

Distribution $\mathcal{P}_\ell$: Choosing $\overrightarrow{T} \sim \mathcal{P}_\ell$ is done according to the following steps.

- Choose s uniformly at random from the range $[k/4, k/2]$. Let $t = 2^s$, that is, t can take $\frac{\log \ell}{4}$ values in the range $[\ell^{1/4}, \ell^{1/2}]$.
- For an $\frac{\ell}{2t} \times \frac{\ell}{2t}$ torus H, choose a random orientation $\overrightarrow{H}$ of H according to the distribution $\mathcal{C}^{(k)}_{\ell/2t}$.
- Set $\overrightarrow{T}'$ to be the t-tiling of $\overrightarrow{H}$.
- Choose $a, b \in [\ell]$ uniformly at random, and set $\overrightarrow{T}$ to be the (a, b)-shifting of $\overrightarrow{T}'$ (see Definition 9.1.3).

Distribution $\mathcal{F}_\ell$: Choosing $\overrightarrow{T} \sim \mathcal{F}_\ell$ is done according to the following steps.

- Choose s uniformly at random from the range $[k/4, k/2]$ and set $t = 2^s$.
- For an $\frac{\ell}{2t} \times \frac{\ell}{2t}$ torus H, choose a random orientation $\overrightarrow{H}$ of H according to the distribution $\mathcal{R}_{\ell/2t}$.
- Set $\overrightarrow{T}'$ to be the t-tiling of $\overrightarrow{H}$.
- Choose $a, b \in [\ell]$ uniformly at random, and set $\overrightarrow{T}$ to be the (a, b)-shifting of $\overrightarrow{T}'$.

9.1.4 Bounding the variation distance

Let $T = (V, E)$ be an $\ell \times \ell$ torus. According to Lemma 9.1.8, every orientation $\overrightarrow{T} \sim \mathcal{P}_\ell$ of T is Eulerian. According to Lemma 9.1.9, an orientation $\overrightarrow{T} \sim \mathcal{F}_\ell$ of T is $\frac{1}{64}$-far from being Eulerian with high probability. Our aim is to show that any non-adaptive deterministic algorithm that makes $o\left(\sqrt{\frac{\log \ell}{\log\log \ell}}\right)$ queries will fail to distinguish between the orientations that are distributed according to $\mathcal{P}_\ell$ and those that are distributed according to $\mathcal{F}_\ell$.

Let $Q \subseteq E$ be a fixed set of at most $\frac{1}{10}\sqrt{\frac{\log \ell}{\log\log \ell}}$ edges queried by a non-adaptive deterministic algorithm. Let $\overrightarrow{H} = (V(H), \overrightarrow{E}(H))$ be the $\frac{\ell}{2t} \times \frac{\ell}{2t}$ torus (oriented according to either $\mathcal{C}^{(k)}_{\ell/2t}$ or $\mathcal{R}_{\ell/2t}$) that has been used to create an orientation $\overrightarrow{T}$ of T. By Definition 9.1.7, the orientations of the padding edges of $\overrightarrow{T}$ are identical in $\mathcal{P}_\ell$ and $\mathcal{F}_\ell$, and the orientations of the other edges are determined by those of their

originating edges. We thus focus on the set $\text{org}(Q) \stackrel{\text{def}}{=} \{\text{org}(e) : e \in Q\} \subseteq E(H)$ of Q's originating edges.

If $\overrightarrow{T}$ is distributed according to $\mathcal{F}_\ell$, then the set $\text{org}(Q) \subseteq E(H)$ is independent in H, and hence the distribution of the orientations of the edges in $\text{org}(Q) \subseteq E(H)$ is uniform. We henceforth assume that $\overrightarrow{T}$ is distributed according to $\mathcal{P}_\ell$. In the next lemmas we show that, with probability at least $4/5$, the set $\text{org}(Q)$ is independent in H also in this case, and thus our algorithm cannot distinguish between the two distributions.

Lemma 9.1.10. *Let $e_1, e_2 \in E$ be two edges within distance x. Let $\overrightarrow{T}$ be a random orientation of T chosen according to the distribution $\mathcal{P}_\ell$. Then*

$$\Pr_s\left[\frac{t}{4k} \leq x \leq 4tk\right] < \frac{8(\log\log\ell + 2)}{\log\ell}.$$

Proof. Observe that there are at most $2\log k + 4 = 2\log\log\ell + 4$ values of s for which a fixed number x can satisfy $2^{s-\log k-2} = \frac{t}{4k} \leq x \leq 4tk = 2^{s+\log k+2}$. Moreover, s is distributed uniformly among its $\frac{k}{4} = \frac{\log\ell}{4}$ possible values, so the lemma follows. □

For a set $Q \subseteq E$ and an orientation $\overrightarrow{T}$ of T, let I_Q be the following event: For all pairs $e_1, e_2 \in Q$, either $\text{dist}(e_1, e_2) < \frac{t}{4k}$ or $\text{dist}(e_1, e_2) > 4tk$ in T.

Lemma 9.1.11. *Let $Q \subseteq E$ be a fixed set of $\frac{1}{10}\sqrt{\frac{\log\ell}{\log\log\ell}}$ edges, and let $\overrightarrow{T}$ be a random orientation of T, according to distribution $\mathcal{P}_\ell$. Then the event I_Q occurs with probability at least $9/10$.*

Proof. Follows by applying the union bound on the inequality from Lemma 9.1.10 for all pairs $e_1, e_2 \in Q$. □

Lemma 9.1.12. *With probability $1 - o(1)$ conditioned on the event I_Q, every two edges $e_1, e_2 \in Q$ such that $\text{dist}(e_1, e_2) < \frac{t}{4k}$ satisfy one of the following: (1) e_1, e_2 are perpendicular; (2) at least one of e_1, e_2 has no origin in H; (3) $\text{org}(e_1) = \text{org}(e_2)$.*

Proof. Fix two edges $e_1, e_2 \in Q$ such that $\text{dist}(e_1, e_2) < \frac{t}{4k}$. If one of them has no originating edge in H then we are done. Otherwise, since $\text{dist}(e_1, e_2) < \frac{t}{4k}$, by the definition of the t-tiling, $\text{org}(e_1)$ and $\text{org}(e_2)$ must have a common endpoint in H, say $v_{i,j}$. If e_1 and e_2 are perpendicular, then again we are done. On the other hand, if e_1 and e_2 are parallel, then in order to have different origins they must be separated

by the the main diagonal of $R_{i,j}$ (the representative grid of the common vertex $v_{i,j}$). Note that this may happen only if the distance of both e_1 and e_2 from the main diagonal is at most $\frac{t}{4k}$. But the probability that an edge is within that distance from the main diagonal of some representative grid is at most $2\ \frac{t}{4k}\ \frac{t}{t^2} = \frac{1}{2k} = o\left(\frac{\log\log \ell}{\log \ell}\right)$. Now the proof is completed by applying the union bound for all pairs $e_1, e_2 \in Q$. $\square$

Lemma 9.1.13. *Let $Q \subseteq E$ be a fixed set of $\frac{1}{10}\sqrt{\frac{\log \ell}{\log\log \ell}}$ edges, and let $\overrightarrow{T}$ be a random orientation of T, chosen according to the distribution $\mathcal{P}_\ell$. Then* $\mathrm{org}(Q)$ *is independent with probability at least* $4/5$.

Proof. By Lemma 9.1.11, with probability at least $9/10$, the event I_Q happens, that is, for all pairs $e_1, e_2 \in Q$ we have either $\mathrm{dist}(e_1, e_2) < \frac{t}{4k}$ or $\mathrm{dist}(e_1, e_2) > 4tk$ in T. Assume that I_Q occurs. Then, by Lemma 9.1.12, with probability $1 - o(1)$, all the pairs $e_1, e_2 \in Q$ with $\mathrm{dist}(e_1, e_2) < \frac{t}{4k}$ in T are perpendicular or have no more than one originating edge. Conditioned on this event, every two edges in $\mathrm{org}(Q)$ are perpendicular or are at distance larger than $2k$ in H. Recall that $|Q| = o(\sqrt{\log \ell}) = o(\sqrt{k})$ and hence $|\mathrm{org}(Q)| = o(\sqrt{k})$. Therefore, from Lemma 9.1.6, $\mathrm{org}(Q)$ is independent in this case with probability $1 - o(1)$.

Summing up, we have that $\mathrm{org}(Q)$ is independent with probability at least $\frac{9}{10} - o(1) > 4/5$ for ℓ large enough, where the probabilities are taken over $\mathcal{P}_\ell$. $\square$

Proof of Theorem 9.1.1. Let $Q \subseteq E$ be the fixed set of $\frac{1}{10}\sqrt{\frac{\log \ell}{\log\log \ell}}$ edges queried by a deterministic non-adaptive algorithm. For every fixed t, a, and b, let $\mathcal{P}_\ell^{t,a,b}$ be $\mathcal{P}_\ell$ conditioned on t, a, and b and let $\mathcal{F}_\ell^{t,a,b}$ be $\mathcal{F}_\ell$ conditioned on t, a, and b. Note that t, a, and b fully define the set $\mathrm{org}(Q)$ of originating edges and, in particular, it is the same set for orientations drawn according to $\mathcal{P}_\ell^{t,a,b}$ and according to $\mathcal{F}_\ell^{t,a,b}$. It follows that, for every t, a, and b, if $\mathrm{org}(Q)$ is independent then the restriction of $\mathcal{P}_\ell^{t,a,b}$ to Q is identical to that of $\mathcal{F}_\ell^{t,a,b}$. Now, recall that for every t, a, and b, if $\overrightarrow{T}$ is distributed according to $\mathcal{F}_\ell^{t,a,b}$ then $\mathrm{org}(Q)$ is independent. On the other hand, by Lemma 9.1.13, $\mathrm{org}(Q)$ is independent with probability at least $4/5$ also for $\mathcal{P}_\ell^{t,a,b}$, taken over the choice of t, a, and b. Summing over all the possible choices of t, a, and b, we obtain that the variation distance between $\mathcal{P}_\ell^{t,a,b}$ and $\mathcal{F}_\ell^{t,a,b}$ is at most $1/5$. Hence, distinguishing between the two distributions with probability larger than $1/5$ requires more than $\frac{1}{10}\sqrt{\frac{\log \ell}{\log\log \ell}} = \Omega\left(\sqrt{\frac{\log m}{\log\log m}}\right)$ queries. $\square$

9.2 A 1-sided lower bound

In this section we prove the following theorem.

Theorem 9.2.1. *For every $0 < \varepsilon \leq 1/16$, every non-adaptive 1-sided ε-test for Eulerian orientations of bounded degree graphs must use at least $\frac{1}{100}m^{1/4}$ queries. Consequently, every adaptive 1-sided test requires $\Omega(\log m)$ queries.*

As opposed to 2-sided testers, a 1-sided tester is not allowed to reject the input unless a negative witness was found. In our case, as claimed in Lemma 7.4.1, the only possible witness that an orientation is not Eulerian is an invalid cut, i.e. a (possibly partial) cut that cannot be made balanced under any orientation of the non-queried edges.

Following this observation, we prove Theorem 9.2.1 using the distribution $\mathcal{F}_\ell$ defined in Subsection 9.1.3. First, we define a distribution $\mathcal{F}'_\ell$ that is similar to the distribution $\mathcal{F}_\ell$, except that t is fixed to be $\ell/16$, and the orientation $\overrightarrow{H}$ of an 8×8 torus H is fixed to be one that makes all 64 vertices fully unbalanced. Then we show that for orientations that are distributed according to $\mathcal{F}'_\ell$, any non-adaptive deterministic algorithm that makes $o(\sqrt{\ell}) = o(\sqrt{m})$ orientation queries cannot find an invalid cut (a negative witness) with probability larger than $1/5$. This will imply that there exists an $\frac{1}{16}$-far orientation on which any randomized tester fails with probability at least $4/5$.

The main idea is as follows. A cut can be invalid (and hence unbalanced) only if both its components contain unbalanced vertices. Let us now fix a cut (A, B) of an $\ell \times \ell$ torus $T = (V, E)$, and let $\overrightarrow{T}$ be an orientation of T chosen according to $\mathcal{F}'_\ell$. Suppose that indeed both A and B contain unbalanced vertices, and let Q be a subset of the edges in the cut (A, B) that witness its invalidity. Using basic properties of tori, we show that either Q contains $\Omega(m^{1/4})$ edges, or otherwise, one of the edges $e \in Q$ must be within distance at most $O(m^{1/4})$ from one of the unbalanced vertices of $\overrightarrow{T}$. Since the number of unbalanced vertices in $\overrightarrow{T} \sim \mathcal{F}'_\ell$ is $O(\ell) = o(m^{1/4})$, and since they are grouped into 64 diagonals of length $\ell/32$, the number of edges that are within distance $O(m^{1/4})$ from these unbalanced vertices is bounded by $O(m^{3/4})$. Finally, since the last step in building $\overrightarrow{T}$ is a random shift, the probability that a set Q of size $o(m^{1/4})$ contains any such edge tends to zero.

We first give a formal definition of the distribution $\mathcal{F}'_\ell$ of orientations over a torus.

Distribution $\mathcal{F}'_\ell$: Choosing $\overrightarrow{T} \sim \mathcal{F}'_\ell$ is done according to the following steps.

- Set $t = \ell/16$.
- Fix the orientation $\overrightarrow{H}$ of the $\frac{\ell}{2t} \times \frac{\ell}{2t} = 8 \times 8$ torus H, such that all 64 vertices of H are fully unbalanced in $\overrightarrow{H}$ (i.e. no vertex has both incoming and outgoing edges).
- Set $\overrightarrow{T}'$ to be the t-tiling of $\overrightarrow{H}$.
- Pick $a, b \in [\ell]$ uniformly at random and set $\overrightarrow{T}$ to be the (a, b)-shifting of $\overrightarrow{T}'$.

Lemma 9.2.2. *Let $T = (V, E)$ be an $\ell \times \ell$ torus and let $\overrightarrow{T}$ be an orientation of T distributed according to $\mathcal{F}'_\ell$. Let Q be a fixed set of $\frac{1}{100}m^{1/4}$ edges from E. Then the probability (over $\overrightarrow{T} \sim \mathcal{F}'_\ell$) that any of the edges in Q is within distance at most $\sqrt{\ell}$ from a vertex $v \in V$ that is unbalanced in $\overrightarrow{T}$ is at most $1/5$.*

Proof. Let U denote the set of unbalanced vertices in $\overrightarrow{T} \sim \mathcal{F}'_\ell$. Observe that $|U| = 64t = 4\ell = 4\sqrt{m/2}$, and recall that the vertices in $U \subseteq V$ are grouped into 64 diagonals of length t (see Definition 9.1.7). Thus, the number of vertices $v \in V$ that are within distance at most $\sqrt{\ell}$ from some vertex $u \in U$ is bounded by $64 \cdot (t + 2\sqrt{\ell}) \cdot 2\sqrt{\ell} \leq 10\ell^{3/2}$. Hence, the probability of a single edge $e \in Q$ satisfying $\text{dist}(e, u) \leq \sqrt{\ell}$ for some $u \in U$ is bounded by $20m^{-1/4}$ and the lemma follows. □

We establish the proof of Theorem 9.2.1 using a few lemmas, in which we point out some significant properties of the torus. But first, we give a general lemma about witnesses for not being Eulerian.

Lemma 9.2.3. *Let $G = (V, E)$ be a graph and let $\overrightarrow{G} = (V, \overrightarrow{E})$ be an orientation of G. If a set $Q \subseteq E$ is a witness that $\overrightarrow{G}$ is not Eulerian then Q contains more than half of the edges of some invalid cut (A, B) in $\overrightarrow{G}$, where both A and B are connected sets of vertices.*

Proof. Recall that, by Lemma 7.4.1, Q contains more than half of the edges of an invalid cut, say (A', B'). Without loss of generality we assume that $|\overrightarrow{E}(A', B')| > \frac{1}{2}|E(A', B')|$. Hence, Q contains more than $\frac{1}{2}|E(A', B')|$ edges going from A' to B'. Let $A_1, \ldots, A_r$ be the connected components of A'. Note that (A', B') is a disjoint union of $(A_1, B'), \ldots, (A_k, B')$. Using averaging calculations, we obtain that there exists a connected component A_i such that Q contains more than $\frac{1}{2}|E(A_i, B')|$ edges

going from A_i to B'. Note in addition that there are no edges between A_i and other connected components A_k's of A', and thus $(A_i, B') = (A_i, V \setminus A_i)$. We conclude that Q contains more than $\frac{1}{2}|E(A_i, V \setminus A_i)|$ edges going from A_i to $V \setminus A_i$. Now, let $B_1, \ldots, B_s$ be the connected components of $V \setminus A_i$. Note that $(A_i, V \setminus A_i)$ is a disjoint union of $(A_i, B_1), \ldots, (A_i, B_s)$. Using averaging calculations, we obtain that there exists a connected component B_j such that Q contains more than $\frac{1}{2}|E(A_i, B_j)|$ edges going from A_i to B_j. Note in addition that there are no edges between B_j and other connected components B_k's of $V \setminus A_i$, and therefore $(A_i, B_j) = (V \setminus B_j, B_j)$. We conclude that Q contains more than $\frac{1}{2}|E(V \setminus B_j, B_j)|$ edges going from $V \setminus B_j$ to B_j, and hence, Q is a witness to the invalidity of $(V \setminus B_j, B_j)$. B_j is clearly connected. To complete the proof, we need to show that $V \setminus B_j$ is connected as well. Recall that $V \setminus B_j$ is a union of A_i and all the connected components B_k's of $V \setminus A_i$ for $k \neq j$. The B_k's are not connected to each other. However, the torus T is a connected graph, and therefore, every B_k must be connected to A_i. Since A_i is connected, $V \setminus B_j$ is connected. To conclude, we set $A = V \setminus B_j$ and $B = B_j$. □

In the following, we let $T = (V, E)$ be an $\ell \times \ell$ torus and use the notation of Definition 9.1.2. For every $i \in [\ell]$, define the i^{th} *row* of T as $R_i \stackrel{\text{def}}{=} \{v_{i,j} \in V \mid j \in [\ell]\}$. For every $j \in [\ell]$, define the j^{th} *column* of T as $C_j \stackrel{\text{def}}{=} \{v_{i,j} \in V \mid i \in [\ell]\}$. Given a set $A \subseteq V$, let $R(A)$ be the set of rows R_i of T such that $A \cap R_i \neq \emptyset$, and let $C(A)$ be the set of columns C_j of T such that $A \cap C_j \neq \emptyset$.

Given a cut (A, B) of V we say that a row R_i is *mixed* if $R_i \subseteq R(A) \cap R(B)$, that is, if R_i includes vertices in A as well as vertices in B. Similarly, we say that a column C_j is *mixed* if $C_j \subseteq C(A) \cap C(B)$. Let r_{mix} be the number of mixed rows with respect to (A, B) and let c_{mix} be the number of mixed columns with respect to (A, B).

Observation 9.2.4. $|E(A, B)| \geq 2(r_{mix} + c_{mix})$.

Proof. Looking at the cycle of vertical edges connecting all the vertices in every mixed column, it is easy to see that every mixed column has at least two vertical edges in (A, B). Similarly, it can be shown that every mixed row has at least two horizontal edges in (A, B). □

Observation 9.2.5.

1. *If* $|R(A)| < \ell$ *then* $c_{mix} = |C(A)|$.

2. *If* $|C(A)| < \ell$ *then* $r_{mix} = |R(A)|$.

The analogous claims also hold for B.

Proof. We give the proof of the first item as the proof of the second item is identical. Let R_i be a row of T that is not in $R(A)$. Then $v_{i,j} \in B$ for every $j \in [\ell]$. Hence, every column $C_j \in C(A)$ has a vertex in A as well as a vertex in B (namely, $v_{i,j}$), which proves the claim. □

Observation 9.2.6.

1. *If* $|R(A)| = \ell$ *then* $r_{mix} = |R(B)|$.

2. *If* $|C(A)| = \ell$ *then* $c_{mix} = |C(B)|$.

Proof. We give the proof of the first item as the proof of the second item is identical. Suppose that $|R(A)| = \ell$. Then every row includes a vertex in A. Let $R_i \in R(B)$. Then R_i includes a vertex in A as well as a vertex in B, which completes the proof. □

We say that a set $A \subseteq V$ of vertices in T is *grid-bounded* if $|R(A)| < \ell$ and $|C(A)| < \ell$.

Lemma 9.2.7. *Let* (A, B) *be a cut of* V *where* $|E(A, B)| < 2\ell$. *Then at least one of* A *and* B *is grid-bounded.*

Proof. From Observation 9.2.4 we have that $r_{mix} + c_{mix} < \ell$. Note that $|R(A)| + |R(B)| - r_{mix} = \ell$ and $|C(A)| + |C(B)| - c_{mix} = \ell$. Hence $|R(A)| + |C(A)| + |R(B)| + |C(B)| \leq 2\ell + r_{mix} + c_{mix} < 3\ell$, and thus, at most two of the sets $R(A), C(A), R(B), C(B)$ are of size ℓ. Assuming that both A and B are not grid-bounded, we have $\max(|R(A)|, |C(A)|) = \ell$ and $\max(|R(B)|, |C(B)|) = \ell$. Therefore, we must have $|R(A)| = \ell$ and $|C(A)| < \ell$ or $|R(A)| < \ell$ and $|C(A)| = \ell$. We complete the proof for the case where $|R(A)| = \ell$ and $|C(A)| < \ell$ as the proof for the other case is identical. From Observation 9.2.6 we have $|R(B)| = r_{mix} < \ell$, and thus, from Observation 9.2.5, $c_{mix} = |C(B)|$. Now, $|C(B)| = \ell$, as otherwise B is grid-bounded. We hence obtain $c_{mix} = \ell$, a contradiction. □

Observation 9.2.8. *If* A *is connected then there exists a row index* $i^* \in [\ell]$ *such that* $R(A) = \{R_{i^*}, R_{i^* \oplus 1}, \ldots, R_{i \oplus (s-1)}\}$ *where* $s = |R(A)|$, *and there exists a column index*

$j^* \in [\ell]$ such that $C(A) = \{C_{j^*}, C_{j^* \oplus 1}, \ldots, C_{j \oplus (t-1)}\}$ where $t = |R(B)|$. Hence, A is contained in a subgraph G of T which is an $|R(A)| \times |R(B)|$ grid. Renaming i^* as 1 and j^* to 1, we have that G is a grid with the vertex set $V_G = \{v_{i,j} \mid i \in [s], j \in [t]\}$.

Proof. Let R_{i_1}, R_{i_2} be rows in $R(A)$. Hence, both R_{i_1} and R_{i_2} include at least one vertex in A. Since A is connected, there exists a path of vertices in A between R_{i_1} and R_{i_2}. Clearly, for every edge in the path, the endpoints are in the same row (in case of a horizontal edge) or in subsequent rows (in case of a vertical edge). We thus conclude that $R(A)$ is a set of successive rows in the torus. Similarly, $C(A)$ is a set of successive columns in the torus. □

Lemma 9.2.9. *Let $T = (V, E)$ be an $\ell \times \ell$ torus, and let $\overrightarrow{T}$ be a non-Eulerian orientation of T. Let $U \subseteq V$ denote the set of unbalanced vertices with respect to $\overrightarrow{T}$. Let $Q \subseteq E$ be a set of edges forming a witness that $\overrightarrow{T}$ is not Eulerian, where $|Q| < \frac{1}{2}\ell$. Let q denote the minimal distance of an edge in Q to an unbalanced vertex, that is, $q \stackrel{\text{def}}{=} \min_{e \in Q,\ u \in U}\{\text{dist}(e, u)\}$. Then $|Q| \geq q$.*

Proof. By Lemma 9.2.3, we assume without loss of generality that Q contains more than half of the edges of an invalid cut (A, B), where both A and B are connected. Since $|Q| < \frac{1}{2}\ell$, we have $|E(A, B)| < \ell$, and hence, from Lemma 9.2.7, one of the sets A and B is grid-bounded. Assume without loss of generality that A is grid-bounded. Let $s = |R(A)|$ and $t = |C(A)|$. Then $s, t < \ell$. Since A is connected, from Observation 9.2.8, A is contained in an $s \times t$ grid G.

Suppose that $|Q| < q$. Then $|E(A, B)| < 2q$, and from Observation 9.2.4 we have $r_{mix} + c_{mix} < q$. Let $e = (w_A, w_B)$ be an edge in $Q \cap E(A, B)$, where $w_A \in A$ and $w_B \in B$. Since A is invalid, there exists an unbalanced vertex $u \in A$. By the definition of q we have $\text{dist}(e, u) \geq q$ and hence $\text{dist}(u, w_A) \geq q$. Using the notation of Observation 9.2.8, we denote $u = v_{i_1, j_1}$ and $w_A = v_{i_2, j_2}$, where $i_1, i_2 \in [s]$ and $j_1, j_2 \in [t]$. Then clearly $|i_1 - i_2| + |j_1 - j_2| \geq q$. We thus have $s = |R(A)| \geq |i_1 - i_2| + 1$ and $t = |C(A)| \geq |j_1 - j_2| + 1$. Since A is grid-bounded, from Observation 9.2.5 we obtain $|C(A)| + |R(A)| = r_{mix} + c_{mix} \geq |i_1 - i_2| + 1 + |j_1 - j_2| + 1 > q$. Finally, Observation 9.2.4 gives that $|E(A, B)| > 2q$, a contradiction. □

Proof of Theorem 9.2.1. Let $T = (V, E)$ be an $\ell \times \ell$ torus and let $\overrightarrow{T} \sim \mathcal{F}'_\ell$ be an orientation of T. Let $Q \subseteq E$ be the fixed set of $\frac{1}{100}m^{1/4}$ edge queries that a deterministic non-adaptive algorithm makes on $\overrightarrow{T}$. By Lemma 9.2.9, in order to form a witness that $\overrightarrow{T}$ is not Eulerian, one of the edges in Q must be within distance

at most $|Q| = \frac{1}{100}m^{1/4} < \sqrt{\ell}$ from an unbalanced vertex in $\overrightarrow{T}$. But according to Lemma 9.2.2, the probability of the above is at most 1/5. We thus conclude that discovering a witness that $\overrightarrow{T} \sim \mathcal{F}'_\ell$ is not Eulerian with probability larger than 1/5 requires more than $\frac{1}{100}m^{1/4}$ nonadaptive queries. $\square$

Part III

Bibliography

Bibliography

[1] N. Alon, E. Fischer, M. Krivelevich and M. Szegedy, Efficient testing of large graphs, *Combinatorica* 20(4):451–476, 2000 (a preliminary version appeared in *Proceedings of the* 40^{th} *FOCS*: 656–666, 1999).

[2] N. Alon, E. Fischer, I. Newman and A. Shapira, A combinatorial characterization of the testable graph properties: It's all about regularity, *Proceedings of the* 38^{th} *STOC*: 251–260, 2006.

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

[4] N. Alon and A. Shapira, Testing subgraphs in directed graphs, *J. Comput. Syst. Sci.* 69(3):354–382, 2004 (a preliminary version appeared in *Proceedings of the* 35^{th} *STOC*: 700–709, 2003).

[5] S. Arora, C. Lund, R, Motwani, M. Sudan and M. Szegedy, Proof verification and the hardness of approximation problems, *Journal of the ACM*, 45(3):501–555, 1998.

[6] L. Babai, On the diameter of Eulerian orientations of graphs, *Proceedings of the* 17^{th} *SODA*: 822–831, 2006.

[7] M. Bender and D. Ron, Testing properties of directed graphs: Acyclicity and connectivity, *Random Structures and Algorithms* 20(2):184–205, 2002.

[8] E. Ben-Sasson, P. Harsha, O. Lachish and A. Matsliah, Sound 3-query PCPPs are Long, *Proccedings of the* 35^{th} *ICALP* (1):686-697, 2008.

[9] E. Ben-Sasson, P. Harsha and S. Raskhodnikova, Some 3CNF properties are hard to test, *SIAM Journal on Computing* 35(1):1–21, 2005 (a preliminary version appeared in *Proceedings of the* 35^{th} *STOC*, 2003).

[10] 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).

[11] G. R. Brightwell and P. Winkler, Counting Eulerian circuits is #P-complete, *Proceedings of the 7th ALENEX and 2nd ANALCO (Vancouver BC)*, C. Demetrescu, R. Sedgewick and R. Tamassia, eds, SIAM Press, 259–262, 2005.

[12] 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.

[13] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron and A. Samorodnitsky, Improved testing algorithms for monotonicity, *Proceedings of the* 3^{rd} *RANDOM*: 97–108, Springer-Verlag, Berkeley CA, 1999.

[14] F. Ergün, S. Kannan, S. R. Kumar, R. Rubinfeld and M. Viswanathan, Spot checkers, *Journal of Computer and System Sciences*, 60(3):717–751, 2000.

[15] E. Fischer, Testing graphs for colorability properties, *Random Structures and Algorithms* 26:289–309, 2005 (a preliminary version appeared in *Proceedings of the* 12^{th} *SODA*, 2001).

[16] 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.

[17] 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.

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

[19] E. Fischer and I. Newman, Testing of matrix-poset properties, *Combinatorica*, to appear (a preliminary version appeared in *Proceedings of the* 33^{rd} *ACM STOC*: 286–295, 2001).

[20] Eldar Fischer and Eyal Rozenberg: Lower bounds for testing forbidden induced substructures in bipartite-graph-like combinatorial objects, *Proceedings of the* 10^{th} *APPROX and* 11^{th} *RANDOM*: 464–478, 2007.

[21] E. Fischer and O. Yahalom, Testing convexity properties of tree colorings, *Proceedings of the* 24^{th} *STACS*, LNCS 4393: 109–120, Springer-Verlag, 2007.

[22] H. Fleishcner, *Eulerian graphs and related topics*, Part 1. Vol. 1. Annals of Discrete Mathematics 45, 1990.

[23] H. Fleishcner, *Eulerian graphs and related topics*, Part 1. Vol. 2. Annals of Discrete Mathematics 50, 1991.

[24] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson, Self-testing/correcting for polynomials and for aproximate functions, *Proceedings of the* 23^{th} *STOC*: 32–42, 1991.

[25] 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.

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

[27] O. Goldreich and L. Trevisan, Three theorems regarding testing graph properties, *Random Structures and Algorithms* 23(1):23–57, 2003 (a preliminary version appeared in *Proceedings of the* 42^{nd} *IEEE FOCS*: 460–469, 2001).

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

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

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

[31] 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.

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

[33] T. Ibaraki, A. V. Karzanov and H. Nagamochi, A fast algorithm for finding a maximum free multiflow in an inner Eulerian network and some generalizations, *Combinatorica 18(1)* (1988), 61–83.

[34] T. Kaufman, M. Krivelevich and D. Ron, Tight bounds for testing bipartiteness in general graphs, *SIAM Journal on Computing* 33(6):1441–1483, 2004.

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

[36] L. Lovász, On some connectivity properties of Eulerian graphs, *Acta Math. Hung.* 28:129–138, 1976.

[37] F. Magniez and M. de Rougemont, Property testing of regular tree languages, In *Proceedings of the* 31^{st} *ICALP*, LNCS 3142: 932–944, Springer-Verlag, 2004.

[38] M. Mihail, P. Winkler, On the number of Eulerian orientations of a graph, *Algorithmica 16(4/5)*: 402–414, 1996.

[39] S. Moran and S. Snir, Convex recolorings of phylogenetic trees: definitions, hardness results and algorithms, to appear in *Journal of Computer and System Sciences* (a preliminary version appeares in *Proccedings of WADS*:218–232, 2005).

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

[41] 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.

[42] 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).

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

[44] P. A. Pevzner, H. Tang and M. S. Waterman, An Eulerian path approach to DNA fragment assembly, *Proc. Natl. Acad. Sci. USA* 98:9748–9753, 2001.

[45] R. W, Robinson, Enumeration of Euler graphs, In *Proof Techniques in Graph Theory* (Ed. F. Harary): 147–153, New York Academic Press, 1969.

[46] 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.

[47] R. Rubinfeld and M. Sudan, Robust characterization of polynomials with applications to program testing, *SIAM Journal on Computing* 25:252–271, 1996 (first appeared as a technical report, Cornell University, 1993).

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

[49] T. Schlieder and F. Naumann. Approximate tree embedding for querying XML data, In *ACM SIGIR Workshop On XML and Information Retrieval*, Athens, Greece, July 2000.

[50] L. Segoufin and V. Vianu, Validating streaming XML documents, *Symposium on Principles of Database Systems (PODS)*:53–64 , 2002

[51] C. Semple and M. Steel, *Phylogenetics*, Oxforx University Press, 2003.

[52] W. T. Tutte, *Graph theory*, Addison-Wesley, New York, 1984.

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