Testing Depth First Search Numbering

Artur Czumaj
Department of Computer Science and DIMAP, University of Warwick, Coventry, UK
University of Cologne, Germany

Christian Sohler
University of Cologne, Germany

Stefan Walzer
Karlsruhe Institute of Technology, Germany

Abstract

Property Testing is a formal framework to study the computational power and complexity of sampling from combinatorial objects. A central goal in standard graph property testing is to understand which graph properties are testable with sublinear query complexity. Here, a graph property P is testable with a sublinear query complexity if there is an algorithm that makes a sublinear number of queries to the input graph and accepts with probability at least 2/3, if the graph has property P, and rejects with probability at least 2/3 if it is ε-far from every graph that has property P.

In this paper, we introduce a new variant of the bounded degree graph model. In this variant, in addition to the standard representation of a bounded degree graph, we assume that every vertex v has a unique label $\texttt{num}(v)$ from $\{1, \ldots, |V|\}$, and in addition to the standard queries in the bounded degree graph model, we also allow a property testing algorithm to query for the label of a vertex (but *not* for a vertex with a given label).

Our new model is motivated by certain graph processes such as a DFS traversal, which assign consecutive numbers (labels) to the vertices of the graph. We want to study which of these numberings can be tested in sublinear time. As a first step in understanding such a model, we develop a *property testing algorithm for discovery times of a DFS traversal* with query complexity $O(n^{1/3}/\varepsilon)$ and for constant $\varepsilon > 0$ we give a matching lower bound.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Randomized Algorithms, Graph Algorithms, Property Testing

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.78

Related Version *Full Version*: https://arxiv.org/abs/2509.05132

Funding *Artur Czumaj*: Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP), by a Weizmann-UK Making Connections Grant, and by an IBM Award. *Stefan Walzer*: Research funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 465963632.

1 Introduction

Depth-first search (DFS) is one of the most useful and frequently used algorithmic primitives in graph algorithms. The DFS algorithm is known for well over a century [17, 23] as a technique for threading mazes and has been widely used in the design of graph and network algorithms since the late 1950s. DFS is a basic tool to traverse a graph in a structured way and is central to solve textbook problems such as connectivity, topological sorting [22], determining strongly connected components in directed graphs [21], biconnected components in undirected graphs [21], and to test planarity [15]. Because of its versatility and usefulness for solving many graph problems, DFS has become one of the most important tools in the design of algorithms for graphs, taught in the first year of many computing science study

programs. One of the most useful combinatorial properties of DFS is the *DFS numbering of vertices*, which is *the order in which a DFS algorithm discovers all vertices of the input graph* [21]. (See also Algorithms 1 and 2, where $N(v)$ denotes the set of neighbors of vertex v.)

Algorithm 1 DFS numbering of G.

Input: undirected graph $G = (V, E)$
Output: $\mathtt{num} : V \to \{1, \ldots, |V|\}$

```
1 mark all vertices as undiscovered
2 t ← 1 // smallest unused ID
3 for v ∈ V do // arbitrary order
4     if v is undiscovered then
        DSFVisit(v)
```

Algorithm 2 DSFVisit(v).

```
1 mark v as discovered
2 num(v) ← t
3 t ← t + 1
4 for u ∈ N(v) do // arbitrary order
5     if v is undiscovered then
        DSFVisit(v)
```

We remark that sometimes, e.g., in the widely used textbook by Cormen et. al. (see Chapter 20.3 in [5]), the DFS numbering is also used together with finishing numbers. We also remark that the labeling is not unique and depends on the ordering in which the vertices are traversed in the for-loops of Algorithms 1 and 2.

▶ **Definition 1** (DFS numberings). *Let $G = (V, E)$ be a labeled undirected graph on n vertices. A labeling $\mathtt{num} : V \to \{1, \ldots, n\}$ is called a* DFS numbering of G *if DFS gives rise to it for some order of processing the vertices in the for-loops of Algorithms 1 and 2.*

Given a wide applicability of DFS and DFS numbering, a natural problem is to verify whether a given graph is correctly DFS-numbered, i.e., the labels assigned to the vertices are a DFS numbering for some ordering of vertices and neighbors. The latter problem is easy to solve by simulating a DFS using the given numbering and locally verifying that the DFS is performed correctly. However, is it also possible to *approximately* verify whether the graph is properly DFS-numbered using a sampling based algorithm? An answer to this question sheds some light on how the local structure of a DFS-numbered graph (as implicitly given by the sample distribution) is related to the global DFS numbering.

We will study this question in the *framework of Property Testing* (see, e.g., the monographs [4, 9]). Property Testing provides a formal setting to study approximate decision problems. The goal is to distinguish objects that satisfy the tested property from those that are ε-far from every object that satisfies the property. Here, ε-far means that one has to modify more than an ε-fraction of the object's description to obtain an object that has the property. A property testing algorithm requires some form of sampling access to the object.

In this paper we will introduce a variant of the standard bounded-degree graph model [12] that in addition to the standard setting, allows also *label queries*. We will assume that there exists some labeling that assigns unique labels from the set $\{1, \ldots, |V|\}$ to the vertex set V, i.e., the labeling is a bijection $\mathtt{num} : V \to \{1, \ldots, |V|\}$. We say that a graph with maximum degree d is ε-far from a DFS-numbered graph, if we have to modify (insert or delete) more than $\varepsilon|V|$ edges to obtain a graph that is correctly DFS-numbered[1]. We do not allow to modify labels, though we notice that allowing to modify labels would also be a valid model (see Section 2.1 for some discussion). Our motivation to not consider it here was merely to stick as closely to the standard testing model as possible.

[1] While one often uses the bound of $\varepsilon d|V|$ edges instead of $\varepsilon|V|$ edges, in the setting when $d = O(1)$ these terms differ only by a constant factor and as such there are essentially indistinguishable.

In this new framework we study a fundamental problem of *whether the input labeled undirected graph is properly DFS-numbered* or *it is ε-far from having a valid DFS numbering*. Our main result is a tight (for constant d and ε) complexity bound for this task. First, we show that any tester for the DFS numbering requires $\Omega(n^{1/3})$ queries, and then, we complement this bound with an algorithm that tests DFS numbering with $O(n^{1/3}/\varepsilon)$ queries.

The proof of the lower bound (Theorem 7 in Section 4) is by constructing two families $(G_n)_{n\in\mathbb{N}}$ and $(B_n)_{n\in\mathbb{N}}$ of *good* and *bad* random labeled graphs for which we will show that distinguishing between these families is necessary for any DFS-tester. Then we will show that distinguishing between these families requires $\Omega(n^{1/3})$ queries.

The proof of the upper bound (Theorem 11 in Section 5) relies on a characterization of labelings that are ε-far from a valid DFS numbering. The characterization is described in a form of some conflicts between the labelings and we design two algorithms detecting such conflicts: one for local conflicts and one for global conflicts. By combining these two algorithms we will obtain an algorithm that tests DFS numbering with $O(n^{1/3}/\varepsilon)$ queries.

1.1 Related work

Property Testing was introduced by Rubinfeld and Sudan [20] and first studied in the dense graph setting by Goldreich and Ron [10]. Constant time testability in the dense graph model has been fairly well understood [2] and is closely related to the regularity lemma. In our paper we build upon the bounded degree graph model, as introduced by Goldreich and Ron [12]. First results in this model included testers for properties such as connectivity, k-connectivity and being Eulerian [12]; bipartiteness [11] and cycle freeness [6] can be tested in the bounded degree graph model in $O(\frac{\sqrt{n}}{\varepsilon^{O(1)}})$ time. A series of papers proved that every hyperfinite property of bounded degree graphs is testable in constant time [3, 7, 14, 18]. Also, every constant time testable property in bounded degree graphs is either finite or contains a hyperfinite property [8]. Furthermore, it is known that every first-order logic property of bounded degree graph with an $\exists\forall$ quantification is constant time testable while some properties with a $\forall\exists$ quantification are not [1]. Further works in the bounded degree graph model include, e.g., testability using proximity oblivious testers [13].

2 Preliminaries: The model

In this paper, let $G = (V, E)$ denote an undirected graph with $n = |V|$ vertices and $m = |E|$ edges. Let $N(v)$ denote the set of *neighbors of* v, i.e., $N(v) := \{u \in V : \{v, u\} \in E\}$, and d be an upper bound on the maximum number of edges incident to any vertex in V. We will assume that G is a labeled graph, i.e., there is a bijection (permutation) $\mathtt{num} : V \to [n]$ that assigns numbers to the vertices of G and we use $[n] := \{1, \ldots, n\}$.

2.1 Bounded degree graph model for *labeled* graphs

In this section we describe how our algorithm can access the input graph $G = (V, E)$ of maximum degree d with vertex labeling $\mathtt{num}$. Our model is a slight modification of the standard bounded degree graph model [12] that in addition *allows access to labels*. As usual in this model, we assume that $V = [n]$ and n as well as d are given to the algorithm. The algorithm can ask the following two types of queries and receives an answer in constant time:

- **Neighbor queries:** for every vertex $v \in V$ and every $1 \le i \le d$, one can query the ith neighbor of vertex v.
- **Label queries:** for every vertex $v \in V$, one can query the label of v, $\mathtt{num}(v)$.

Observe that we allow access to vertices and their neighbors, and we can check the label $\mathtt{num}(v)$ of any vertex v, ***but we have no access to vertices through their labels*** $\mathtt{num}$. In particular, *to access vertex v with a given label* $\mathtt{num}$*, we have to query the oracle until such vertex is returned.* This is a special feature distinguishing our model from the standard model used in graph property testing and making the problem challenging: we have a labeled graph, but we cannot access the graph (its vertices) through the labels! Instead, the only way to access the graph is by taking any vertex $v \in V = [n]$ and either querying its label $\mathtt{num}(v)$ (via the label query) or querying its ith neighbor (via a neighbor query).

The *query complexity* of a testing algorithm is the number of oracle queries.

A property testing algorithm (in short, *property tester*) for DFS-numbered graphs is an algorithm that has access to an input graph as described above and that accepts the input with probability at least 2/3, if G is a graph with $\mathtt{num}$ being a valid DFS-numbering. The algorithm rejects G with probability at least 2/3 if G is ε-far from having a valid DFS numbering $\mathtt{num}$ according to the following definition. (The algorithm developed in this paper is a *property tester with one-sided error*, i.e., it always accepts, if $\mathtt{num}$ is a DFS-numbering.)

▶ **Definition 2** (ε-far from DFS numberings in bounded degree graphs)**.** *Let $G = (V, E)$ be an undirected graph on n vertices with maximum degree at most d and let $\mathtt{num} : V \to [n]$ be a bijection that assigns labels to V. We say the labeled graph G $\mathtt{num}$* is ε-far from having a valid DFS numbering*, if one has to modify[2] more than εn edges in G to obtain a graph $G' = (V, E')$ of maximum degree at most d for which $\mathtt{num}$ is a valid DFS numbering.*

Implicitly labeled graphs. To simplify the presentation, we will often assume $\mathtt{num}(v) = v$. However, *we will not use this knowledge in the algorithm* as the model prevents us from querying $\mathtt{num}^{-1}$. Our tester will only ask for a random vertex or for a vertex that occurred as the neighbor of a previously queried vertex.

Further thoughts about the model: Modifying the labels. Observe that in general, it might be natural to consider a revised definition of a labeling $\mathtt{num}$ being ε-far from a valid DFS numbering, where while defining graph G' in Definition 2, in addition to edge modifications, one would allow also for modifications of the labels[3]. We observe that for bounded degree graphs, adding the labels modifications does not change the problem significantly.

▶ **Lemma 3.** *Let $\mathtt{num}$ be a permutation of V to $\{1, \ldots, n\}$. For a given bounded degree graph $G = (V, E)$, a labeling $\mathtt{num}$ is ε-far from a valid DFS numbering according to the edges modifications if and only if $\mathtt{num}$ is $\Theta(\varepsilon)$-far from a valid DFS numbering according to the edges and labels modifications.*

Proof. Observe that the number of modifications of the edges to obtain a valid DFS numbering is not smaller than the number of modifications of the edges and the labels to obtain a valid DFS numbering. Therefore, if for a given bounded degree graph G, a labeling $\mathtt{num}$ is ε-far from a valid DFS numbering according to the edges and labels modifications then $\mathtt{num}$ is also ε-far from a valid DFS numbering according to the edges modifications.

---

[2] Modification of the edges means edge insertions and deletions, i.e., we require $|E \triangle E'| > \varepsilon n$, where $\triangle$ is the symmetric difference.

[3] We use the following revised definition: a labeling $\mathtt{num}$ *is ε-far from a valid DFS numbering* of G if for any graph $G' = (V, E')$ of maximum degree at most d with a valid DFS numbering $\mathtt{num}'$ we have $|E \triangle E'| + |\{v \in V : \mathtt{num}(v) \neq \mathtt{num}'(v)\}| \geq \varepsilon n$.

For the other direction, if num is ε-far from a valid DFS numbering according to the edges modifications then we can simulate modification of the labels by modification of the edges: to assign a given label to a vertex we just remove all its incident edges and add all edges incident to the vertex with the label sought. (Observe that a vertex might have had some edges of its own that have to be removed but by correcting the labels one by one, we can ensure that once we fix vertex v by assigning it to label i with vertex u having label i before, then later we will have to fix the label of vertex u, resolving the issue.) Observe that if we apply this operation to all vertices with the labels to be changed, then we do not increase the maximum degree, and hence, modifying of k labels can be simulated by modifying at most $2dk$ edges. Therefore, if a labeling num is ε-far from a valid DFS numbering according to the edges modifications then num is also $(2d\varepsilon)$-far from a valid DFS numbering according to the edges and labels modifications. ◀

Why should the labels be a permutation? Our model assumes that the input labeling num is a permutation (bijection) of V to $\{1 \dots, n\}$, but it may be natural to consider the case when num is *not necessarily a permutation* but rather an arbitrary function from V to $\{1 \dots, n\}$, allowing repetitions of labels. Observe that already the simple problem of testing if num is a permutation or is ε-far from being a permutation (also known as the element distinctness problem) is known to require $\Theta(\sqrt{n}/\varepsilon)$ queries (see, e.g., [19]). In view of that bound, the best what we could hope for without assuming that num is a permutation of V to $\{1 \dots, n\}$ is to achieve query complexity of $\Theta(\sqrt{n}/\varepsilon)$. And this can be easily obtained by combining our algorithm in Theorem 11 with the known algorithms testing element distinctness, leading to a testing algorithm with $\Theta(\sqrt{n}/\varepsilon)$ queries.

3 Basic Properties of DFS Numberings

We begin with a review of basic DFS terminology, introduce some notions of our own, and make a few useful observations. (We also refer to standard textbooks, e.g., [5, 16].)

The DFS algorithm, as given in Algorithm 1, numbers every vertex even if G is *disconnected* due to the outer loop over all vertices. Every vertex is initially *undiscovered*. When DSFVisit(v) is called, v becomes *discovered*. We say v is *active* as long as the call DSFVisit(v) persists and is *finished* as soon as it ends. The vertex p that initiates the call of DSFVisit(v) is the *parent* of v. If the call of DSFVisit(v) is initiated from the outer loop then we say that v is an *orphan* with *virtual parent* 0. The idea is that the outer loop iterating over all vertices is like a call to DSFVisit(0) where 0 is a virtual vertex connected to all other vertices. Each connected component of G has exactly one orphan, which receives the smallest number in its connected component. At any point during the execution the *white path* consists of all active vertices including the virtual vertex 0, with every active vertex (other than 0) connected to its parent. The DFS numbers appear in increasing order along the white path. In an implicitly labeled graph $G = (V, E)$, we define $p(v)$ for $v \in V$ as

$$p(v) := \begin{cases} \max\{u \in N(v) \cap [v-1]\} & \text{if } N(v) \cap [v-1] \neq \emptyset \\ 0 & \text{otherwise.} \end{cases},$$

▷ Claim 4. If the numbers in an implicitly labeled graph correspond to a DFS numbering then $p(v)$ is the (possibly virtual) parent of v.

Proof. If v is an orphan then it has the smallest number in its connected component so $\{u \in N(v) : u < v\} = \emptyset$. Hence $p(v) = 0$, which is the virtual parent of orphans by definition.

Now assume v was discovered from a non-virtual parent $x \in V$. Then $x \in N(v)$ with $x < v$ so $x \in \{u \in N(v) : u < v\}$. To see that x is the maximum of $\{u \in N(v) : u < v\}$, consider $x' \in \{x+1, \dots, v-1\}$. Immediately before v is discovered, x is at the end of the white path, hence the largest active vertex. Since $x' > x$ has been discovered, x' must be finished. Hence all neighbours of x' have been discovered. Since v is not discovered we have $x \notin N(v)$ so $x \notin \{u \in N(v) : u < v\}$. ◁

Observe that any edge $e = \{u, v\}$ with $u < v$ in an implicitly labeled graph corresponds to an *interval* $[u, v]$ representing a range of elements with respect to the DFS numbering. The analysis of the inter-relation between such intervals plays a central role in our analysis.

▶ **Definition 5.** *A pair* $(v, \{u, w\}) \in V \times E$ *is a* ***conflicting pair*** *if* $p(v) < u < v < w$.

The following central lemma shows that the absence of conflicting pairs is necessary and sufficient for the validity of a DFS numbering (we defer a simple proof to the full version).

▶ **Lemma 6.** *Let* $G = (V, E)$ *be an implicitly labeled graph. The following are equivalent.*

(i) *G has a valid DFS numbering.*

(ii) *There exists no conflicting pair.*

4 Testing DFS Numbering Requires $\Omega(n^{1/3})$ Queries

In this section, we prove our first main result.

▶ **Theorem 7.** *Every property tester for the property of having a valid DFS-labeling has a query complexity of* $\Omega(n^{1/3})$.

Let $\varepsilon > 0$ be sufficiently small (any $\varepsilon \leq \frac{1}{33}$ would do). The proof of Theorem 7 is by constructing two families $(G_n)_{n\in\mathbb{N}}$ and $(B_n)_{n\in\mathbb{N}}$ of *good* and *bad* random labeled graphs for which we will show that distinguishing between these families is necessary for any DFS-tester. Then we will show that distinguishing between these families requires $\Omega(n^{1/3})$ queries.

4.1 Construction of good and bad random labeled graphs (G_n) and (B_n)

Let $n, N \in \mathbb{N}$. Each graph G_n and B_n consists of $\lfloor \frac{n}{8N} \rfloor$ *arms* of $8N$ vertices each. The roots of these arms are connected with some binary tree that is the same for G_n and B_n.

There are four types of arms, as described in details in Figure 1. Each arm of G_n is a copy of $(G1)$ or $(G2)$ chosen independently and uniformly at random. Similarly, each arm of B_n is a copy of $(B1)$ or $(B2)$ chosen independently and uniformly at random.

The labeling of G_n is then obtained by starting in the root of the tree T and using within the arms the relative ordering defined by the numbering of the arms $(G1)$ or $(G2)$. For an example, see, e.g., Figure 3. (Since each arm has the same number of vertices, the choice of one arm, whether in $(G1)$ or $(G2)$, does not affect the numbering of other arms.)

Similarly (see Figure 4), the labeling of B_n is defined by starting at a root of T and using within the arms the relative ordering defined by the numbering of the arms $(B1)$ or $(B2)$.

4.2 Properties of DFS numberings of random labeled graphs G_n and B_n

Let us first notice that our construction of good random labeled graphs $(G_n)_{n\in\mathbb{N}}$ easily ensures that they have valid DFS numberings (with probability 1).

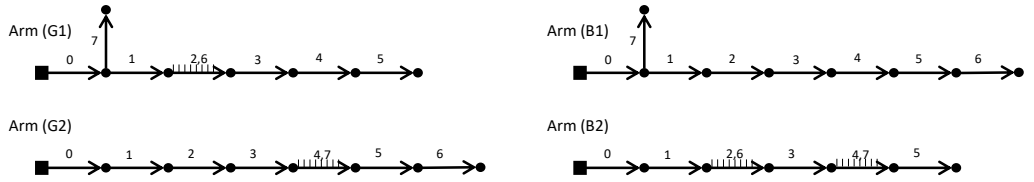
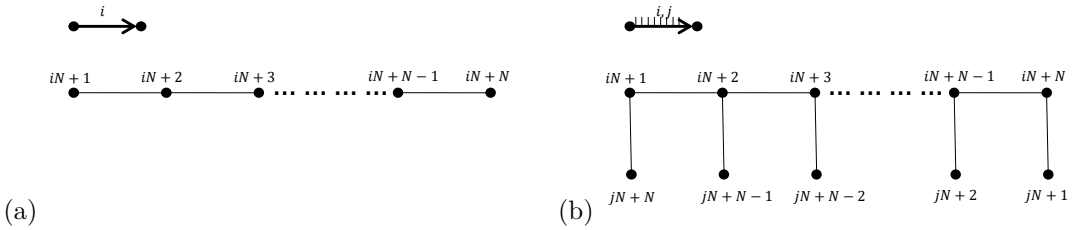


Figure 1 Four types of *arms* $(G1)$, $(G2)$, $(B1)$, and $(B2)$ used in our construction of $(G_n)_{n\in\mathbb{N}}$ and $(B_n)_{n\in\mathbb{N}}$. Each arm starts with a root, denoted by ■. Each link $\xrightarrow{\ i\ }$ corresponds to a path on N vertices and labels $iN+1,\ldots,i(N+1)$ ascending in the direction of the arrow (see Figure 2(a)). We also have a *comb graph* $\xrightarrow{\ i,j\ }$ (see Figure 2(b)), which is obtained from $\xrightarrow{\ i\ }$ by adding N new vertices, adding a matching between the new vertices and the vertices from $\xrightarrow{\ i\ }$, and labeling the new vertices with $jN+1,\ldots,(j+1)N$, this time descending in the direction of the arrow (i.e., the vertex with label $iN+k$ is adjacent to the vertex with label $jN+N-k+1$).



(a) (b)

Figure 2 Graphs corresponding to (a) a path $\xrightarrow{\ i\ }$ and (b) a comb graph $\xrightarrow{\ i,j\ }$.

▶ **Lemma 8.** *G_n has a valid DFS numbering.* ◀

A situation is different for bad random labeled graphs $(B_n)_{n\in\mathbb{N}}$. While the arms of type $(B1)$ also locally maintain a valid DFS-order, the arms of type $(B2)$ do not (because of the two comb graphs). As the result, the construction of $(B_n)_{n\in\mathbb{N}}$ typically gives labelings that are ε-far from valid DFS numberings.

▶ **Lemma 9.** *Let $0 \le \varepsilon \le \frac{1}{33}$ and let n be sufficiently large. Then B_n has a labeling that is ε-far from a valid DFS numbering with probability $1-o(1)$.*

Proof. Consider the numbering for $(B2)$. Let us fix any $k \in \{1,2,\ldots,N\}$ and focus on vertices p_1, c_1, p_2, c_2 with labels $2N+k, 7N-k+1, 4N+k, 8N-k+1$, respectively. (For example, in Figure 4 in the top branch (where numbers have an offset of 1), if we took $k=2$ then we would have $\langle p_1, c_1, p_2, c_2\rangle$ to be vertices with labels $\langle 11, 28, 19, 32\rangle$ in B_{64}.) Observe that the fact that $\mathtt{num}(p_1) < \mathtt{num}(p_2) < \mathtt{num}(c_1) < \mathtt{num}(c_2)$ implies that this is not a valid DFS numbering as long as c_1 is the DFS-child of p_1 and c_2 is the DFS-child of p_2. Therefore, to turn the labeled graph into one consistent with DFS numbering, one has to add or delete at least one edge incident to a vertex from $\{p_1, c_1, p_2, c_2\}$. Since such a quadruple is obtained for each $k \in \{1,\ldots,N\}$ and since each of these quadruples is disjoint (for example, in Figure 4, the there are four such quadruples formed by vertices with labels $\langle 10, 29, 18, 33\rangle$, $\langle 11, 28, 19, 32\rangle$, $\langle 12, 27, 20, 31\rangle$, and $\langle 13, 26, 21, 30\rangle$), one needs to modify at least $\frac{N}{2}$ edges to obtain a valid DFS numbering. Since the expected number of copies of $(B2)$ in B_n is $\frac{1}{2}\cdot\lfloor\frac{n}{8N}\rfloor$, the expected number of changes required in B_n to obtain a valid DFS numbering is at least $\frac{N}{4}\cdot\lfloor\frac{n}{8N}\rfloor$. For sufficiently large n, a standard Chernoff bound implies that graph B_n is $\frac{1}{33}$-far from having a valid DFS numbering with high probability. ◀

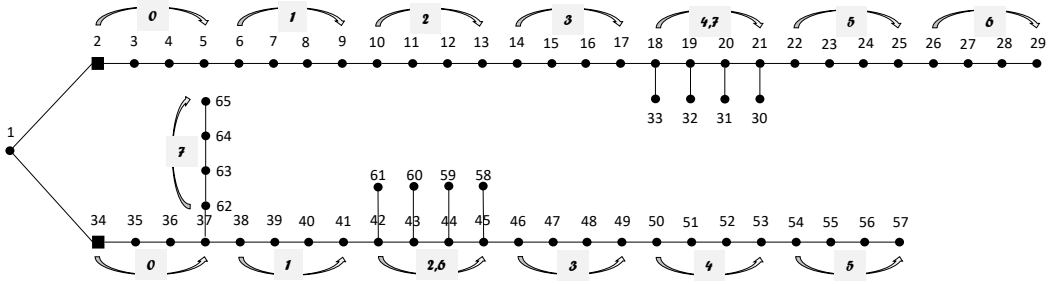


Figure 3 An example of labeling of G_n with $n = 64$, $N = 4$, using $\lfloor \frac{n}{8N} \rfloor = 2$ arms of size $8 \cdot 4 = 32$. Vertices with labels 1, 2, 34 are in the tree T, the top branch (with labels 2–33) corresponds to arm $(G2)$ and the bottom branch (with labels 34–65) corresponds to arm $(G1)$. For each arm, we also marked the corresponding parts defining it.

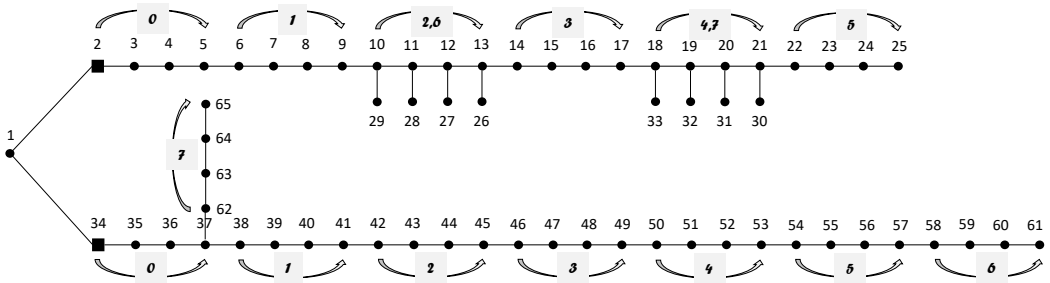


Figure 4 An example of labeling of B_n with $n = 64$, $N = 4$, using $\lfloor \frac{n}{8N} \rfloor = 2$ arms of size $8 \cdot 4 = 32$. Vertices with labels 1, 2, 34 are in the tree T, the top branch (with labels 2–33) corresponds to arm $(B2)$ and the bottom branch (with labels 34–65) corresponds to arm $(B1)$. For each arm, we also marked the corresponding parts defining it.

4.3 Hardness of distinguishing between good and bad labeled graphs

Our next central lemma provides a lower bound for the number of queries of any algorithm ALG that distinguishes between the families of random labeled graphs $(G_n)_{n\in\mathbb{N}}$ and $(B_n)_{n\in\mathbb{N}}$. We will assume that the input I_n for $n \in \mathbb{N}$ on which ALG requires q_n oracle queries is obtained by first selecting a random bit $b \in \{0, 1\}$ and then setting $I_n = \begin{cases} G_n & \text{if } b = 0, \\ B_n & \text{if } b = 1. \end{cases}$

▶ **Lemma 10.** *Let $0 < \xi \leq \frac{N^3}{n}$. For every randomized algorithm ALG' that receives I_n as input, performs $q_n \leq \sqrt{\frac{\xi n}{4N}}$ queries, and outputs a bit $b' \in \{0, 1\}$, we have $\Pr[b = b'] \leq \frac{1}{2} + \xi$.*

Proof. Let ALG' be an algorithm that receives I_n as its input. We assume that ALG' can query the oracle for any vertex u by submitting an ID of u, and the oracle returns the label $\mathfrak{num}(u)$ of u, and the IDs of all neighboring vertices. Further, without loss of generality, we assume that the IDs $1, \ldots, n$ are randomly assigned to the vertices of the graph. Therefore ALG' is limited to querying for a random vertex (a *random query*) or for a vertex that has been previously found as a neighbor of an earlier queried vertex (an *explorative query*).

Without loss of generality, we can assume that the vertices selected by random queries are chosen in the order $v_1, v_2, \ldots$ before the run of ALG'; let $V_R = \{v_1, \ldots, v_{q_n}\}$ be the sequence of these first q_n random vertices (ALG can select only these vertices). Let $\mathcal{E}$ be the random

event that no two vertices $u, u' \in V_R$ come from the same arm of I_n. We prove the following:

$$\Pr[\neg\mathcal{E}] \leq \xi \tag{1}$$
$$\Pr[b = b'|\mathcal{E}] = \tfrac{1}{2} \tag{2}$$

Observe that the two inequalities (1)–(2) imply Lemma 10:

$$\Pr[b = b'] = \Pr[b = b'|\mathcal{E}] \cdot \Pr[\mathcal{E}] + \Pr[b = b'|\neg\mathcal{E}] \cdot \Pr[\neg\mathcal{E}] \leq \tfrac{1}{2} \cdot 1 + 1 \cdot \xi \leq \tfrac{1}{2} + \xi \ .$$

In order to prove inequality (1), let us first define C to be the number of pairs $i \neq j$ with $i, j \leq q_n$ such that both v_i and v_j come from the same arm of I_n, or in other words, so that v_i and v_j belong to the same copy of arm $(B1)$. We have,

$$\mathbb{E}[C] = \sum_{1 \leq i < j \leq q_n} \Pr[v_i \text{ and } v_j \text{ are in the same arm}] \leq \binom{q_n}{2} \cdot \frac{8N}{n} \leq \frac{4q_n^2 N}{n} \leq \frac{4(\frac{\xi n}{4N})N}{n} \leq \xi \ .$$

Therefore we can conclude (1) by Markov's inequality: $\Pr[\neg\mathcal{E}] = \Pr[C \geq 1] \leq \mathbb{E}[C] \leq \xi$.

Next, we want to prove inequality (2). Let V_R^* be the set of vertices reachable from vertices in V_R in at most q_n steps, that is, $V_R^* = \{u : \exists_{1 \leq i \leq q_n} \operatorname{dist}(v_i, u) \leq q_n\}$. Let $I_n(V_R^*)$ be the subgraph of I_n induced by the vertex set V_R^*. We observe that ALG' will make its decision solely on seeing some subgraph of $I_n(V_R^*)$. Hence, the output b' of ALG' is a (random) function of $I_n(V_R^*)$. Now, we claim that conditioned on $\mathcal{E}$, the random variables b and $I_n(V_R^*)$ are stochastically independent, which in turn, would imply identity (2).

To prove that b and $I_n(V_R^*)$ are independent, let us consider a single vertex v_i from V_R and the subgraph $I_n(v_i)$ induced by vertices with distance at most q_n from v_i. If $I_n(v_i)$ contains at least one vertex from T, then $I_n(v_i)$ consists solely of some of the vertices from T and some vertices that are close to the roots of some of the arms (within parts denoted by $\xrightarrow{0}$ of the arms, since each such part has length N and we have $q_n \leq \frac{N}{2}$ since $q_n \leq \sqrt{\frac{\xi n}{4N}}$ and $\xi \leq \frac{N^3}{n}$). Since these parts are identical in G_n and B_n, such paths share no information on b.

Otherwise, if $I_n(v_i)$ contains no vertex from T, then $I_n(v_i)$ is a part of an arm of I_n. But then, due to $q_n \leq \frac{N}{2}$, at most one joint[4] of this arm is contained in $I_n(v_i)$. Since the labels of the roots of the arms are the same in G_n and B_n, it is always well-defined what the offset of a label within its arm is. For more details, see the full version of the paper. ◀

4.4 Hardness of testing DFS numbering (proof of Theorem 7)

By Lemmas 8 and 9, any algorithm ALG that accepts a labeled graph with a valid DFS numbering with probability at least $\frac{2}{3}$ and rejects a labeled graph with a DFS numbering that is ε-far (for $0 < \varepsilon \leq \frac{1}{33}$) from being valid DFS numbering with probability at least $\frac{2}{3}$, must be able to distinguish with probability at least $\frac{5}{8}$ between the families of good and bad random labeled graphs $(G_n)_{n \in \mathbb{N}}$ and $(B_n)_{n \in \mathbb{N}}$. However by Lemma 10, by setting $\xi = \frac{1}{8}$ and $N = \lfloor n^{1/3} \rfloor$, the tasks of distinguishing between these families requires $\Omega(n^{1/3})$ queries. ◀

5 Testing DFS Numbering with $O(n^{1/3}/\varepsilon)$ Queries

Our second main result shows that the lower bound in Theorem 7 is asymptotically tight.

[4] By a *joint* we mean an endpoint of a path or comb from which the arm was stitched together.

▶ **Theorem 11.** *Let $0 < \varepsilon < 1$. There is an algorithm that with oracle access to a labeled bounded degree undirected graph G on n vertices, performs $O(n^{1/3}/\varepsilon)$ queries to the oracle and accepts, if G has a valid DFS numbering, and rejects with probability at least $\frac{2}{3}$, if G is ε-far from having a valid DFS numbering.*

(One can show that our tester achieves also the same $O(n^{1/3}/\varepsilon)$ running time.)

For the rest of the section, let G be a corresponding labeled graph equipped with a possibly inappropriate DFS numbering $\mathtt{num} : V \to [n]$. As in Section 3, we assume that $\mathtt{num}$ *is implicit* so that we can, for instance, write $v < w$ for $v, w \in V$ instead of $\mathtt{num}(v) < \mathtt{num}(w)$.

Outline. Our approach to prove Theorem 11 is first to extend Lemma 6 (which characterizes valid DFS numberings) to describe a simple and useful property of labelings that are ε-far from a valid DFS numbering. In Lemma 13 in Section 5.1, we will show that if the numbering is ε-far from a valid DFS numbering then not only we have $\Omega(\varepsilon n)$ conflicting pairs (in the sense of Lemma 6), but in fact we have $\Omega(\varepsilon n)$ conflicting pairs that are "unrelated." Once we have this property, the task in hand will be to detect any of such conflicting pair. We observe that there are two types of conflicting pairs, *local pairs* involving vertices whose DFS numbers are close to each other, and *global conflicts*. In order to detect local conflicts, we first develop (in Section 5.2) some basic tools to traverse a given graph following DFS numbering. Once we know how to traverse the graph, we can design an algorithm that can determine if a given pair $(v, \{u, w\})$ is conflicting pair. Unfortunately, this algorithm is efficient only if vertex u or w is close to $p(v)$ or v (that is, if one of $\mathtt{num}(u) - \mathtt{num}(p(v))$, $\mathtt{num}(v) - \mathtt{num}(u)$, $\mathtt{num}(w) - \mathtt{num}(v)$ is small), and so we can use this approach to deal with local conflicts. In order to study global conflicts, we notice that if vertices u and w are far away from vertices $p(v)$ and v, then in fact a conflicting pair $(v, \{u, w\})$ can be also extended to multiple vertices v. Once we have that property, we will show that if we sample randomly $\Theta(n^{1/3}/\varepsilon)$ vertices and $\Theta(n^{1/3}/\varepsilon)$ edges, then if there were many global conflicts, then there would be one that is determined by one of the sampled vertices and one of the sampled edges.

5.1 Properties of labelings that are ε-far from any valid DFS numbering

We begin with describing a useful property of labelings that are ε-far from a valid DFS numbering that they have $\Omega(\varepsilon n)$ conflicting pairs that are "unrelated." Recall that by Lemma 6 a numbering is a valid DFS numbering if and only if there is no conflicting pair $(v, \{u, w\}) \in V \times E$ with $p(v) < u < v < w$. In that case, when $p(v) < u < v < w$, we speak of a conflict involving vertex v and edge $\{u, w\}$. While Lemma 6 characterizes valid and invalid DFS numberings, we will need a stronger claim about properties of numberings that are ε-far from valid DFS numberings. For that, we need to understand numberings that are not ε-far from valid DFS numberings because we can modify the input graph with at most εn edge modifications to ensure that the resulting graph will have a valid DFS numbering.

Observe that Lemma 6 provides a simple tool to edit the input labeled graph to obtain a valid DFS numbering – to remove all conflicting pairs. With that in mind, the following claim provides a simple fix to remove all conflicts involving a specific vertex or all conflicts involving a specific edge (notice that the resulting graph may violate our degree bound d, but one can address this issue using a "degree reduction framework," see the full version.

▶ **Lemma 12.** *Let $v \in V$ and $\{u, w\} \in E$ with $u < w$.*

- **(i)** *Adding the edge $\{v-1, v\}$ to G (and doing nothing if $\{v-1, v\}$ is already present) does not create any new conflicts and resolves all conflicts involving v.*
- **(ii)** *Removing $\{u, w\}$ from G and adding $\{w-1, w\}$ (if not already present) does not create any new conflicts and resolves all conflicts involving $\{u, w\}$.*

Proof.

(i) After the edit we have $p(v) = v - 1$ so no $u \in V$ can satisfy $p(v) < u < v$ any more, meaning all conflicts involving v are resolved. We do not create any new conflicts because $p(v-1)$ does not change and the new edge $\{v-1, v\}$ cannot be involved in a conflict because (again) no v' can satisfy $v - 1 < v' < v$.

(ii) Deleting $\{u, w\}$ clearly resolves all conflicts of this edge. However, new conflicts involving w may arise since $p(w)$ may change if we had $p(w) = u$ before. By (i) we can fix these by adding $\{w - 1, w\}$. ◀

Lemma 12 shows that adding or removing a few edges can resolve many related conflicts. We use this claim to show that in order for G to be ε-far from having a valid DFS numbering, not only do there have to be many conflicts, but in fact there have to be *many mutually unrelated conflicts*. To formalize this idea we consider the bipartite *conflict graph* $\mathcal{C} = (V, E, C)$ where $C \subseteq V \times E$ contains an edge $(v, \{u, w\})$ precisely if it is a conflicting pair. The intuition of mutually unrelated conflicts corresponds to a matching in $\mathcal{C}$. We have the following.

▶ **Lemma 13** (ε-far DFS numberings). *If G is ε-far from having a valid DFS numbering then there is a matching $M \subseteq C$ of size $|M| \geq \varepsilon n/5$ in $\mathcal{C}$.*

Proof. We will prove Lemma 13 by showing that if G is ε-far from having a valid DFS numbering then $\mathcal{C}$ has a vertex cover of size at least $\varepsilon n/5$, for if not, then we could modify at most εn edges of G to make the labeling to be a valid DFS numbering of the resulting graph. Then Lemma 13 follows from König's theorem.

Let $M \subseteq C$ be a maximum matching in $\mathcal{C}$. By Kőnig's theorem there is a vertex cover $\mathcal{VC} \subseteq V \cup E$ of size $|\mathcal{VC}| = |M|$, i.e., a set of vertices and edges of G (vertices of $\mathcal{C}$) such that for every conflict pair $(v, \{u, w\})$ (for every edge of $\mathcal{C}$) $v \in \mathcal{VC}$ or $\{u, w\} \in \mathcal{VC}$. We can fix all conflicts in G in $\leq 2|\mathcal{VC}|$ edits by applying for each $v \in \mathcal{VC} \cap V$ the fix of Lemma 12 (i) (one edit) and for each $\{u, w\} \in \mathcal{VC} \cap E$ the fix of Lemma 12 (ii) (two edits). We obtain a graph G^* with a valid DFS numbering. However, the vertices that appeared in a fix in the role of v or w may have degree $d + 1$ in G^*, higher than permitted. By our "degree reduction framework" (see full version), another $3|\mathcal{VC}|$ edits suffices to transform G^* into $G^{**} = (V, E^{**})$ with maximum degree d while maintaining the validity of the DFS numbering. Overall $|E^{**} \triangle E| \leq 5|\mathcal{VC}|$ and since G is ε-far we have $|E^{**} \triangle E| \geq \varepsilon n$. Hence $|M| = |\mathcal{VC}| \geq \varepsilon n/5$. ◀

Lemma 13 immediately implies a simple tester detecting ε-far instances: we randomly sample $\Omega(\sqrt{n/\varepsilon})$ vertices and $\Omega(\sqrt{n/\varepsilon})$ edges, and then with a constant probability one of the sampled vertices v and one of the sampled edges e will form a conflicting pair (v, e).

▶ **Corollary 14.** *There is an algorithm that with oracle access to a labeled bounded degree graph G on n vertices, performs $O(\sqrt{n/\varepsilon})$ queries to the oracle and accepts, if G has a valid DFS numbering, and rejects with probability $\frac{2}{3}$, if G is ε-far from having a valid DFS numbering.* ◀

In what follows, we will show how to improve this result, as promised in Theorem 11.

5.2 Navigating (would-be) DFS trees

In this section, we develop tools to find conflicting pairs for vertices. We first show how to traverse a graph using consecutive labels (a simple proof is deferred to the full version).

▶ **Fact 15.** *Let T be an ordered tree of bounded degree and S its canonical DFS ordering*[5]*. For a randomly selected vertex v from T, given oracle access to T, it is possible to locate the successor (and the predecessor) of v in S, if one exists, with $O(1)$ queries to T in expectation.*

Using Fact 15, we can prove the following lemma.

▶ **Lemma 16** (DFS-NEXT)**.** *There is an algorithm DFS-NEXT that for a given vertex $v \in V$:*

- *If the numbering of G is a valid DFS numbering then either vertex $v+1$ is returned or END-OF-COMPONENT is returned if v is the largest vertex in its connected component.*
- *The expected (over vertices in V) number of queries to G of algorithm DFS-NEXT is $O(1)$.*

There also exists an algorithm DFS-PREV with corresponding properties.

Proof. Let T be the ordered tree on $V \cup \{0\}$ rooted at 0 with the edges $\{p(v), v\}$ for $v \in V$ and children ordered ascending by number. If the numbering of G is a valid DFS numbering then (by Claim 4) T is precisely the DFS tree that gave rise to the numbering. We can navigate to successors and predecessors in T (by Fact 15) with an expected number of $O(1)$ queries to G per step. The only problem is that the virtual vertex 0 cannot be accessed and has unbounded degree. Whenever we would have to access it, we return END-OF-COMPONENT instead, which is appropriate because a valid DFS backtracks to 0 precisely if a connected component has been fully explored. (If the numbering of G is invalid, the produced answer may be meaningless, but it is still obtained within the claimed expected time budget.) ◀

5.3 Testing for conflicts

Lemma 13 implies that an ε-far instance has $\Omega(\varepsilon n)$ pairwise distinct vertices and edges involved in conflicts. In this section, we will extend that approach and study separately *local conflicting pairs* and *global conflicting pairs* in order to improve the tester from Corollary 14. This notion depend on a locality parameter that we will later choose as $\ell = n^{1/3}$.

▶ **Definition 17.** *Let $(v, \{u, w\}) \in V \subseteq E$ with $p(v) < u < v < w$ be a conflicting pair. We speak of a **local conflict** of the following (not mutually exclusive) types* **(L1)** *if $u - p(v) \leq \ell$ and $p(v) \neq 0$,* **(L2)** *if $v - u \leq \ell$,* **(L3)** *if $w - v \leq \ell$. In all other cases we speak of a **global conflict**:* **(G)** *if $p(v) = 0$ and $\max\{v-u, w-v\} > \ell$; or if $\max\{u - p(v), v-u, w-v\} > \ell$.*

Informally, local conflicts occur when u is close (in the sense of its DFS number) to $p(v)$ or v, or w is close v, in which case one can traverse the input graph to efficiently detect the conflict. For global conflicts, some more global approach will be needed.

5.3.1 Testing for local conflicts

We can detect local conflicts by sampling vertices at random and walking forwards and backwards in the (supposed) DFS order using Lemma 16 as follows.

▶ **Lemma 18.** *There is an algorithm that with $O(\ell/\varepsilon)$ queries in expectation accepts all valid DFS numberings and rejects with probability $2/3$ if there is a matching $M \subseteq V \subseteq E$ of size at least $\frac{\varepsilon n}{30}$ consisting of conflicting pairs of type* **(L1)**. *The same applies to types* **(L2)**, **(L3)**.

Proof. Consider the following procedure WALK-FROM-$p(v)$ that is given $v \in V$ as input. It first checks if $p(v) \neq 0$. If so, it attempts to locate vertices $p(v)+1, p(v)+2, \ldots$ using DFS-NEXT from Lemma 16 until one of the following happens.

[5] The DFS starts at the root and respects the order of children when visiting them in pre-order traversal.

- If DFS-NEXT reaches vertices out of order, then report the error.
- If DFS-NEXT reaches a vertex u that has a neighbour $w > v$ then report the error.
- If v or $p(v) + \ell$ is reached without finding an error, then conclude that v is not involved in a conflict of type **(L1)**.

Clearly WALK-FROM-$p(v)$ finds an error if v is involved in a conflict of type **(L1)** and by Lemma 16 it performs $O(\ell)$ queries in expectation (over all $v \in V$). Our algorithm to detect conflicts of type **(L1)** is now simply to repeat WALK-FROM-$p(v)$ for many v sampled uniformly at random. Since the $\frac{\varepsilon n}{30}$ vertices matched in M are pairwise distinct, a random vertex from V is involved in a conflict of type **(L1)** with probability $\frac{\varepsilon}{30}$. If we sample $\frac{60}{\varepsilon}$ vertices at random then the probability of sampling at least one v involved in a conflict of type **(L1)** is

$$\geq 1 - (1 - \varepsilon/30)^{60/\varepsilon} \geq 1 - \mathrm{e}^{-(\varepsilon/30)\cdot(60/\varepsilon)} = 1 - \mathrm{e}^{-2} \geq 2/3 \ .$$

The expected total number of queries amounts to $O(\ell/\varepsilon)$, which concludes the claim.

For conflicts of type **(L2)** a similar procedure WALK-BACKWARDS-FROM-v works. For conflicts of type **(L3)** a procedure WALK-FORWARDS-FROM-v would *not* work because DFS-NEXT might report END-OF-COMPONENT before w is reached (and without us being aware of w's existence). A procedure WALK-BACKWARDS-FROM-w does work, however. Note that we have to sample $\frac{60d}{\varepsilon}$ edges uniformly at random which is still $O(1/\varepsilon)$ because d is constant. ◀

5.3.2 Testing for global conflicts

Lemma 18 provides an efficient tool to detect local conflicts but for global conflicts we use a different approach. We rely on Lemma 13 that promises that there is a matching $M \subseteq V \subseteq E$ of size at least $\varepsilon n/10$ consisting of conflicting pairs. Therefore, if most of conflicts defining the matching M are global, we can use the following lemma.

▶ **Lemma 19.** *There is an algorithm that performs $O(\sqrt{n/\ell}/\varepsilon)$ queries in expectation and accepts all valid DFS numberings and that rejects with probability at least $2/3$ if there is a matching $M \subseteq V \subseteq E$ of size at least $\varepsilon n/10$ consisting of conflicting pairs of type (G).*

Proof. Let us partition matching M into strips $M_1 \oplus M_2 \oplus \cdots \oplus M_{\lceil \frac{n}{\ell} \rceil}$ according to the number of u, so that $M_j = \{(v, \{u, w\}) \in M : \lceil u/\ell \rceil = j\}$. Let us define sets $V_j := \{v \in V : \exists_{e \in E}\ (v, e) \in M_j\}$ and $E_j := \{e \in E : \exists_{v \in V}(v, e) \in M_j\}$. Let $\bar{v}_j$ be the median of V_j. Let V_j^- be those vertices from V_j that are at most $\bar{v}_j$ and E_j^+ those edges from E_j matched to a vertex that is at least $\bar{v}_j$. Let $m_j := |V_j^-|$ and note that $|E_j^+| = m_j$ and that $m_j \geq \frac{|M_j|}{2}$.

We claim that *every pair* in $V_j^- \times E_j^+$ is a conflicting pair. Indeed, let $v_1 \in V_j^-$, $\{u_2, w_2\} \in E_j^+$ with $u_2 < w_2$, and let $p_1 = (v_1, \{u_1, w_1\})$ and $p_2 = (v_2, \{u_2, w_2\})$ be the corresponding pairs in M_j. Then we observe the following:

$$p(v_1) \overset{(1)}{\leq} \max\{0, u_1 - \ell\} \overset{(2)}{<} u_2 \overset{(2)}{<} u_1 + \ell \overset{(1)}{<} v_1 \overset{(3)}{\leq} \bar{v}_j \overset{(3)}{\leq} v_2 \overset{(4)}{<} w_2 \ .$$

To see this, we notice the following:

(1) p_1 is of type (G). Hence $u_1 - p(v_1) > \ell$, unless $p(v_1) = 0$; moreover $v_1 - u_1 > \ell$;
(2) since both p_1 and p_2 are in M_j, we have $|u_1 - u_2| < \ell$;
(3) by the definition of V_j^- and E_j^+ the values of v_1 and v_2 fall on the respective side of the median $\bar{v}_j$;
(4) p_2 is a conflicting pair.

In view of the above, since every pair in $V_j^- \times E_j^+$ is a conflicting pair, we observe that over all j we do not just have $m = |M|$ conflicting pairs to work with, but a collection of bicliques with $\sum_{i=1}^{\lceil n/\ell \rceil} (m_j)^2$ conflicting pairs in total. If we happen to get hold of a $v \in V_j^-$ and an $e \in E_j^+$ for the same j, then we have a conflicting pair (v, e).

In order to find a conflicting pair using the approach above, let us sample (i.u.r.) s vertices from V and then (i.u.r.) s edges from E, where s will be set up momentarily.

For any $1 \leq i, r \leq s$, let $X_{i,r}$ to be the indicator random variable that the ith sampled element from V and the rth sampled element from E are (respectively) from the sets V_j^- and E_j^+ for the same j. Observe that if we define random variable X as $X = \sum_{i=1}^{s} \sum_{r=1}^{s} \mathbb{E}[X_{i,r}]$, then by the arguments above, if X is positive then we have detected a conflicting pair. Using the second moment method, we get prove the following lemma.

▶ **Lemma 20.** *Let $s \geq c\sqrt{\frac{n^3}{\ell \cdot |M|^2}}$ for a sufficiency large constant c and let $s = \omega((d/\varepsilon)^3)$. Then* $\Pr[X > 0] \geq 0.99$.

Hence, if we sample i.u.r. at least $c \cdot \sqrt{\frac{n^3}{\ell |M|^2}}$ vertices from V and at least $c \cdot \sqrt{\frac{n^3}{\ell |M|^2}}$ edges from E for a sufficiently large constant c, then with a constant probability we will detect a conflicting vertex. At the same time, if the DFS numbering is valid then the algorithm will accept it. This yields Lemma 19 since in our setting $|M| = \Omega(\varepsilon n)$ and $\ell = n^{1/3}$. ◀

5.4 Putting all together: the proof of Theorem 11

Now we are ready to complete the proof of Theorem 11. Our algorithm runs the algorithms from Lemma 18 (responsible or conflicts of type **(L1)**, **(L2)** and **(L3)**) and from Lemma 19 one after the other. If any of them rejects G then we reject G, otherwise we accept G. The *expected* total running time is $O(\ell/\varepsilon + \sqrt{n/\ell}/\varepsilon)$, which with $\ell = \Theta(n^{1/3})$ gives $O(n^{1/3}/\varepsilon)$ as claimed. (The query complexity can be made $O(n^{1/3}/\varepsilon)$ using Markov inequality.)

Concerning correctness, it is clear that instances with valid DFS numberings are always accepted. If G is ε-far from a valid DFS numbering then by Lemma 13 there is a matching $M \subseteq V \subseteq E$ of $|M| \geq \varepsilon n/5$ conflicting pairs. Since each matching edge falls into (at least) one type, at least one of the following statements holds.

- There is a matching $M_{\textbf{(L1)}}$ of $|M_{\textbf{(L1)}}| \geq \varepsilon n/30$ conflicting pairs of type **(L1)**.
- There is a matching $M_{\textbf{(L2)}}$ of $|M_{\textbf{(L2)}}| \geq \varepsilon n/30$ conflicting pairs of type **(L2)**.
- There is a matching $M_{\textbf{(L3)}}$ of $|M_{\textbf{(L3)}}| \geq \varepsilon n/30$ conflicting pairs of type **(L3)**.
- There is a matching $M_{\textbf{(G)}}$ of $|M_{\textbf{(G)}}| \geq \varepsilon n/10$ conflicting pairs of type **(G)**.

In each case, the corresponding algorithm rejects G with probability $2/3$ so overall we reject G with probability at least $2/3$. ◀

6 Conclusions

In this paper we introduced a variant of the standard bounded-degree graph model in the property testing setting that works for labeled graphs and allows also label queries. We demonstrated the strength of the model on our new study of DFS numbering. Our main technical contribution is a tight analysis for detecting whether the input labeled graph is properly DFS-numbered or it is ε-far from having a valid DFS numbering. We demonstrated that this task can be solved with $\Omega(n^{1/3}/\varepsilon)$ queries and also $\Omega(n^{1/3})$ queries are necessary.

We observe that while our analysis is presented for undirected graphs, similar arguments hold also for ***directed graphs***. The lower bound from Theorem 7 trivially extends to directed graphs and a careful pass through the algorithm in Theorem 11 shows that the analysis can be extended accordingly. However, to implement this algorithm efficiently in our setting, we need to allow access to incoming and outgoing edges. (See the full version of the paper.)

Our analysis can be also extended (but only for undirected graphs) to the ***DFS finishing numbers*** (FIN-numberings [5]). We can show that (for undirected graphs) a numbering $\mathtt{num} : V \to [n]$ is a valid FIN numbering iff the reverse numbering $\overleftarrow{\mathtt{num}}$ with $\overleftarrow{\mathtt{num}}(i) = n+1-i$ is a valid DFS numbering. This immediately implies that our results for testing valid DFS numberings extend to testing valid FIN numberings in undirected graphs.

References

1 Isolde Adler, Noleen Köhler, and Pan Peng. On testability of first-order properties in bounded-degree graphs and connections to proximity-oblivious testing. *SIAM Journal on Computing*, 53(4):825–883, 2024. doi:10.1137/23M1556253.

2 Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM Journal on Computing*, 39(1):143–167, 2009. doi:10.1137/060667177.

3 Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. *Advances in Mathematics*, 223(6):2200–2218, 2010.

4 Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing – Problems and Techniques*. Springer Verlag, 2022. doi:10.1007/978-981-16-8622-1.

5 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 4th Edition*. MIT Press and McGraw-Hill, 2022.

6 Artur Czumaj, Oded Goldreich, Dana Ron, C. Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 45(2):139–184, 2014. doi:10.1002/RSA.20462.

7 Artur Czumaj, Asaf Shapira, and Christian Sohler. Testing hereditary properties of non-expanding bounded-degree graphs. *SIAM Journal on Computing*, 38(6):2499–2510, 2009. doi:10.1137/070681831.

8 Hendrik Fichtenberger, Pan Peng, and Christian Sohler. Every testable (infinite) property of bounded-degree graphs contains an infinite hyperfinite subproperty. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 714–726, 2019. doi:10.1137/1.9781611975482.45.

9 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

10 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

11 Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bunded degree graphs. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 289–298, 1998. doi:10.1145/276698.276767.

12 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/S00453-001-0078-7.

13 Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40(2):534–566, 2011. doi:10.1137/100789646.

14 Avinatan Hassidim, Jonathan A. Kelner, Huy N. Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 22–31, 2009. doi:10.1109/FOCS.2009.77.

15 John E. Hopcroft and Robert E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852.

16 Jon M. Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, 2013.

17 Édouard Lucas. *Récreations Mathématiques*. Librairie Albert Banchard, Paris, 1882.

18 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013. doi:10.1137/120890946.

19 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam D. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009. doi:10.1137/070701649.

20 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

21 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

22 Robert Endre Tarjan. Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6:171–185, 1976. doi:10.1007/BF00268499.

23 G. Tarry. Le problème des labyrinthes. *Nouvelles Annales de Mathématiques, 3e série*, 14:187–190, 1895.