A Lower Bound on Cycle-Finding in Sparse Digraphs

Xi Chen* Tim Randolph† Rocco A. Servedio‡ Timothy Sun§

Abstract

We consider the problem of finding a cycle in a sparse directed graph G that is promised to be far from acyclic, meaning that the smallest feedback arc set in G is large. We prove an information-theoretic lower bound, showing that for N-vertex graphs with constant outdegree any algorithm for this problem must make $\tilde{\Omega}(N^{5/9})$ queries to an adjacency list representation of G. In the language of property testing, our result is an $\tilde{\Omega}(N^{5/9})$ lower bound on the query complexity of one-sided algorithms for testing whether sparse digraphs with constant outdegree are far from acyclic. This is the first improvement on the $\Omega(\sqrt{N})$ lower bound, implicit in Bender and Ron [BR02], which follows from a simple birthday paradox argument.

1 Introduction

In the current massive data era there is great interest in the abilities and limitations of sublinear time algorithms for various computational problems. In particular, in recent years a number of researchers have studied sublinear time algorithms for fundamental graph problems such as approximating the size of the minimum vertex cover [PR07, MR09, NO08, YYI09, HKNO09, ORRR12], the number of connected components [CRT05], maximum matching [NO08, YYI09], and the minimum spanning tree weight [CRT05, CS09, CEF+05]; counting edges [Fei06, GR08, BHPR+18], stars [GRS11, ABG+18], triangles [ELRS17], k-cliques [ERS18], and arbitrary subgraphs [AKK19]; finding forbidden minors [KSS18, KSS19]; and checking k-colorability [RD85], bipartiteness [GR02], planarity [BSS08], and more. The sublinear time regime imposes natural constraints on algorithms. For instance, a simple "needle in a haystack" lower bound argument shows that it is impossible to distinguish acyclic graphs from graphs with one or more cycles in time sublinear in the number of edges. As a result, sublinear graph algorithms typically provide either approximate guarantees on their output[1] or are designed for property testing-style problems in which the input graph G is promised to satisfy some condition that allows a sublinear algorithm to succeed.[2] Our results are of the second type: *We prove a lower bound on the running time of algorithms for finding cycles in sparse digraphs that are promised to be not too close to acyclic.*

To motivate our inquiry, we observe that many of the most fascinating and enigmatic objects of modern scientific research, such as brains, neural networks, social networks, and the Internet, are naturally modeled as *massive, sparse, directed graphs*. Thus it is a compelling goal to understand the capabilities of sublinear time algorithms on such graphs. Despite this fact, although there is a substantial literature on property testing in general undirected graphs (see Chapters 8, 9, and 10 of [Gol17]), we are aware of fewer works on sublinear time algorithms or property testing on sparse directed graphs [OR11, HS12, HS13, CPS16]. The most directly relevant previous work that we are aware of is the early paper of Bender and Ron [BR02] on testing acyclicity in directed graphs, which we discuss in detail below.

1.1 The query model and promise problem that we consider.

Throughout this work, we consider digraphs on N vertices named $[N] := \{1, \ldots, N\}$ in which the outdegree of each vertex is bounded from above by a small absolute constant d. It suffices to take $d \geq 80$ for our main result to hold. Graphs are represented using the adjacency list model, in which a query consists of a vertex $u \in [N]$ and an index $i \in [d]$. In response, the algorithm receives the i^{th} outneighbor of u or an empty string if u has fewer than i outneighbors. The query complexity of an algorithm is the maximum number of queries that it makes on any N-vertex graph.

The algorithmic problem we consider is that of outputting a directed cycle given an input graph G. The promise that G contains at least one cycle is insufficient to allow sublinear time algorithms: for instance, if G consists of a single constant-length cycle and all other vertices are isolated, or if G consists of a single cycle

*Columbia University. Email: xichen@cs.columbia.edu.
†Columbia University. Email: t.randolph@columbia.edu.
‡Columbia University. Email: rocco@cs.columbia.edu.
§Columbia University. Email: tim@cs.columbia.edu.

[1] For instance, the algorithms of [ORRR12, CRT05, ELRS17, AKK19].

[2] For instance, the algorithms of [KSS18, RD85, GR02, BSS08].

of length N, $\Omega(N)$ queries are required. Hence, in the spirit of property testing, we consider the promise problem in which the input graph G is promised to be ϵ-*far from acyclic*. This means that the smallest *feedback arc set*[3] of G is of size at least ϵdN.

As we discuss later in item (1) of Section 9, the promise that G is ϵ-far from acyclic ensures that G must contain very short cycles [Fox18], so this promise eliminates the concern that merely outputting a directed cycle necessitates $\Omega(N)$ runtime. However, it is far from clear how many queries may be required to *find* a cycle in sparse directed graphs that are ϵ-far from acyclic. This question was implicitly considered by Bender and Ron: in [BR02] they gave an $\Omega(N^{1/3})$-query lower bound on property testing algorithms for testing whether a bounded-degree digraph is acyclic versus ϵ-far from acyclic with two-sided error in the adjacency list model. Implicit in the proof of their $\Omega(N^{1/3})$ lower bound is an $\Omega(N^{1/2})$ lower bound for one-sided testers, or equivalently, for algorithms that find a directed cycle in far-from-acyclic bounded-degree digraphs. We give a proof sketch of this lower bound in Section 3.1; as we explain there, their $\Omega(N^{1/2})$ lower bound is based on a simple birthday paradox argument but such an argument cannot succeed in obtaining an $\omega(N^{1/2})$ lower bound. We note that Bender and Ron [BR02] state as an explicit goal for future work the problem of improving their lower bound, and that acyclicity testing in bounded-degree digraphs is listed as "Open Problem #41" on the website `sublinear.info`.[4]

1.2 Our result: An $\tilde{\Omega}(N^{5/9})$-query lower bound.

Our main result is a proof that any randomized algorithm under the adjacency list query model must make $\tilde{\Omega}(N^{5/9})$ queries to find a cycle in a sparse N-vertex digraph that is ϵ-far from acyclic. The lower bound holds even if ϵ is a fixed constant. In more detail, our main result is the following:

Theorem 1 (Main theorem)**.** *Let d, ϵ be fixed constants with $d \geq 80$ and $\epsilon \leq 1/60$, and let G be an arbitrary digraph, promised to be ϵ-far from acyclic and with outdegree bounded above by d. Any algorithm that, given query access to the adjacency list representation of G, outputs a directed cycle in G with constant probability must make $\tilde{\Omega}(N^{5/9})$ queries.*

We give a detailed discussion of our techniques in Section 3; as explained there, the arguments underlying our lower bound are significantly more involved, both conceptually and technically, than the $\Omega(N^{1/2})$ lower bound of [BR02] for the same problem.

2 Preliminaries

A *directed graph* (or *digraph*) $G = (V, E)$ consists of a set V of vertices and a set E of directed edges. Each edge directed from u to v is represented by the pair (u, v). The *outdegree* (resp. *indegree*) of a vertex u is the number of edges (u, v) (resp. (v, u)) between u and an *outneighbor* (resp. *inneighbor*) $v \in V$. We say a digraph has outdegree bounded by d if every vertex has outdegree at most d. A digraph is ϵ-*far from acyclic* if the minimum feedback arc set has size ϵdN (that is, at least ϵdN edges must be removed to make G acyclic). An *out-tree* is an acyclic digraph in which there exists a unique directed path from a *root* vertex to every other vertex. A vertex in an out-tree with no outgoing edge is called a *leaf*; otherwise it is called an *internal vertex*. An out-tree is said to have degree d if every internal vertex has outdegree exactly d.

Given a positive integer n, we write $[n]$ to denote $\{1, \ldots, n\}$. For fixed d and ϵ, we consider the problem of finding a cycle in a digraph $G = ([N], E)$, with outdegree bounded by d, that is ϵ-far from acyclic. Algorithms may query the *adjacency list representation* of G as follows. We assume the algorithm knows N. A query consists of a vertex $u \in [N]$ and an index $i \in [d]$. In response, the algorithm receives the i^{th} outneighbor of u or an empty string if u has fewer than i neighbors. For convenience, we simplify the adjacency list query model to the *vertex query* model, in which the algorithm simply queries a vertex u and receives an ordered list containing all outneighbors of u. Clearly, algorithms on digraphs with maximum outdegree at most d under the vertex query model can be implemented in the adjacency list model by increasing the number of queries by a factor of d, and thus asymptotic lower bounds in the vertex query model also hold in the adjacency list model.

Throughout the paper, random variables are indicated by a bold font and distributions are indicated by blackboard bold.

3 Our techniques

As is standard in property testing, we employ Yao's principle [Yao77] to prove our lower bound. By this principle, to prove Theorem 1 it suffices to define a probability distribution over N-vertex digraphs with outdegree bounded by d and argue that

1. A random $\boldsymbol{G}$ drawn from this distribution is ϵ-far from acyclic with probability $1 - o_N(1)$.

[3] Recall that a subset $S \subset E$ of directed edges in a graph is a feedback arc set if every directed cycle in G contains at least one edge in S, or equivalently, deleting all the edges in S makes G become acyclic.

[4] https://sublinear.info/index.php?title=Open_Problems:41

2. Any deterministic algorithm $\mathcal{A}$ that makes

$$Q^* := \frac{N^{5/9}}{\log N}$$

queries to $\boldsymbol{G}$ finds a cycle with probability $o_N(1)$.

In this section we first present a simple distribution from [BR02] and sketch the $\Omega(N^{1/2})$ lower bound for this distribution that is implicit in the arguments of [BR02]. We then outline the difficulty inherent in proving an asymptotically better lower bound, informally describe the distribution $\mathbb{BR}$ that we use for Theorem 1, and outline our proof of the theorem.

3.1 A simple $\Omega(N^{1/2})$ lower bound due to Bender and Ron. The distribution over sparse digraphs we now describe corresponds to the distribution $\mathcal{G}_2$ defined in Section 4 of [BR02]; we denote this distribution by $\mathbb{BR}_{\text{simple}} := \mathbb{BR}_{\text{simple}}(N, d)$. A graph $\boldsymbol{G}$ drawn from $\mathbb{BR}_{\text{simple}}$ is generated by randomly partitioning the N vertices $\{1, \ldots, N\}$ into two equal-size subsets $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ and taking d random directed perfect matchings from $\boldsymbol{S}_1$ to $\boldsymbol{S}_2$ and d random directed perfect matchings from $\boldsymbol{S}_2$ to $\boldsymbol{S}_1$ as the edges of $\boldsymbol{G}$.

A straightforward probabilistic analysis (see Lemma 5 of [BR02]) shows that for any constant $d \geq 128$, a random graph $\boldsymbol{G} \sim \mathbb{BR}_{\text{simple}}$ is ϵ-far from acyclic for $\epsilon = 1/16$. To complete the lower bound, it remains to argue that any deterministic algorithm $\mathcal{A}$ that makes $o(N^{1/2})$ queries finds a directed cycle in $\boldsymbol{G} \sim \mathbb{BR}_{\text{simple}}$ with probability $o_N(1)$. This follows from the following stronger property: with probability $1 - o_N(1)$ over the choice of $\boldsymbol{G} \sim \mathbb{BR}_{\text{simple}}$, no deterministic algorithm that makes $o(N^{1/2})$ queries receives in response to a query a vertex it has previously observed, either as input to or output from a query.[5] This property follows from a standard birthday paradox type argument, i.e., the fact that a sequence of $o(N^{1/2})$ uniform samples from an N-element set samples the same element twice with probability $o_N(1)$.

3.2 A challenge in going beyond $N^{1/2}$ many queries. Another birthday paradox type argument demonstrates that a random walk in $\boldsymbol{G} \sim \mathbb{BR}_{\text{simple}}$ will collide with itself in $O(N^{1/2})$ steps with high probability, thus yielding a cycle. Hence a different construction must be considered to obtain an $\omega(N^{1/2})$ lower bound.

The essence of the simple $\Omega(N^{1/2})$ lower bound is that with high probability, the algorithm receives no "useful" information about the underlying graph: each of $o(N^{1/2})$ many queries yields an answer that is uniform over all previously unseen vertices. Unfortunately, this property does *not* hold for algorithms that make $\omega(N^{1/2})$ queries. For example, an algorithm that repeatedly queries (u, i) pairs drawn uniformly at random from $[N] \times [d]$ would observe i.i.d. draws from some distribution over $[N]$. Because $\omega(N^{1/2})$ i.i.d. draws from *any* distribution supported on at most N elements will result in $\omega_N(1)$ collisions with high probability, any argument establishing an $\omega(N^{1/2})$ lower bound must contend with the nontrivial information that algorithms receive about the unknown underlying graph through collisions. Indeed, the central difficulty in proving an $\omega(N^{1/2})$ lower bound is showing that no algorithm can gain enough information from induced collisions to find a cycle.

3.3 Our construction and a sketch of our main ideas. We now give an informal description of the distribution $\mathbb{BR} := \mathbb{BR}(N, d)$ that we analyze (a detailed description is given in Section 4). This distribution is a modified version of a construction proposed by Bender and Ron in [BR02].

Each graph in the support of $\mathbb{BR}$ has $3N$ vertices, and each vertex has outdegree either d or 0. A graph $\boldsymbol{G}$ drawn from $\mathbb{BR}$ is obtained as follows: N vertices are randomly selected and designated as *blue* vertices, and the remaining $2N$ vertices are designated as *red* vertices. Red vertices are randomly partitioned into L many layers $R_1, \ldots, R_L$, each containing $W = 2N/L$ vertices.[6] Each blue vertex is assigned d outneighbors by choosing each one uniformly at random from the blue vertices and the first half of the layers of the red vertices. Each red vertex in layer R_i $(i < L)$ is assigned d outneighbors by choosing each one uniformly from the W vertices in R_{i+1}. For a visual example, refer to Figure 1. A straightforward probabilistic argument (given in Section 4.2) shows that with probability $1 - o_N(1)$ a random $\boldsymbol{G} \sim \mathbb{BR}$ is ϵ-far from acyclic, so the main challenge is to show that it is hard to find a directed cycle in a graph drawn from this distribution.

We give some intuition behind the construction of graphs in $\mathbb{BR}$. Note that every cycle in $\boldsymbol{G}$ consists entirely of blue vertices. Thus, a cycle-finding algorithm may want to "avoid wandering into the red region." This, however, is difficult to do because the local neighborhood of a typical vertex "looks the same" whether it is blue or red (note that an algorithm under the adjacency list model of course never receives explicit

[5] We assume without loss of generality that the algorithm never repeats a previous query.

[6] Both L and W are $N^{\Theta(1)}$; the exact values will be given later and are not important for our intuitive discussion here.

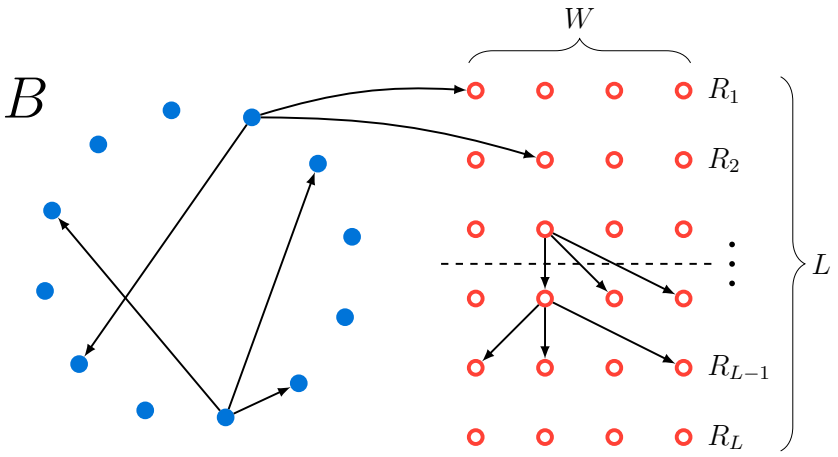

Figure 1: Cartoon of a random graph $\boldsymbol{G} \sim \mathbb{BR}$.

information about whether any particular vertex is blue or red). For example, the simple random walk approach sketched at the beginning of the previous subsection will not work for $\boldsymbol{G} \sim \mathbb{BR}$: even if the random walk starts at a blue vertex, after $O(1)$ steps on average it will reach a red vertex and will have no chance of completing a cycle. Given that an algorithm needs $\Omega(N^{1/2})$ queries to find a cycle even if it is given the set of blue vertices (since the blue part of $\boldsymbol{G} \sim \mathbb{BR}$ is very similar to graphs drawn from $\mathbb{BR}_{\text{simple}}$ described in Section 3.1), it is natural to hope for an $\omega(N^{1/2})$ lower bound using the distribution $\mathbb{BR}$.

There are two challenges in obtaining an $\omega(N^{1/2})$ lower bound using $\mathbb{BR}$. First, as discussed in the previous subsection (which applies not only to $\mathbb{BR}$ but to any distribution), an $\omega(N^{1/2})$-query algorithm may experience many collisions and hence potentially obtain a significant amount of information about $\boldsymbol{G}$. The second challenge is specific to $\mathbb{BR}$. Despite the intuition, "wandering into the red region" may actually provide useful information about $\boldsymbol{G}$ when done strategically (see Section 8 for two attacks on $\mathbb{BR}$ based on exploring the red region; they together imply that one cannot hope to obtain a lower bound better than $N^{13/18}$ using $\mathbb{BR}$). Given that many algorithmic strategies are possible, how can one argue that *every* algorithm that does not make too many queries is unlikely to find a cycle?

To explain the intuition that underlies our lower bound, we first note that for a query on vertex u to reveal a cycle, it must be the case that u is blue and there is a directed path from one of its outneighbors to u in the current "knowledge graph" of the algorithm (where the knowledge graph consists of all edges that have been found so far and v is an ancestor of u if it has a directed path to u). As a result, we focus on the maximum number of ancestors among all blue vertices in the current knowledge graph because the probability that an algorithm discovers a cycle when it queries a vertex is proportional to its number of ancestors in the knowledge graph. Our proof, at a high level, shows that this crucial quantity cannot grow too fast.

A key notion behind our analysis is the division of a sequence of queries made by an algorithm into distinct *epochs*. Roughly speaking, an epoch ends either when a collision occurs (i.e., one of the outneighbors of the vertex queried is a vertex that the algorithm has seen before, either as a query vertex or as an outneighbor of a query vertex), or when "too many" queries have been made since the end of the previous epoch. We introduce the notion of epochs in Section 5 and bound the number of epochs that occur in the execution of any algorithm that makes at most Q^* queries (Lemma 2). We also bound the number of *blue surprise epochs*: these are epochs that end because the vertex u queried is blue and has a blue outneighbor v that the algorithm has seen before (Lemma 3). We pay special attention to such epochs because with the discovery of (u, v), all ancestors of u become ancestors of v and thus, the number of ancestors of v may grow rapidly.

Next, in Section 6 we show that during an epoch of any algorithm, regardless of outcomes of previous epochs, the vertices queried are unlikely to contain a path of blue vertices of length more than $4 \log N$. We do so by analyzing the information that an algorithm has about the connected components of the knowledge graph at any point in its execution, and arguing that the distribution of colors of unqueried children of the knowledge graph is close to a "naive distribution" against which no algorithm can succeed in constructing a long blue path with high probability. We use this

argument to prove Lemma 4, which is at the heart of our lower bound argument. In particular, Lemma 4 implies that during an epoch that is not a blue surprise epoch (or during a blue surprise epoch but ignoring the blue-blue collision edge found at the end of the epoch), the maximum number of ancestors of blue vertices in the knowledge graph can increase by no more than $4 \log N$.

Finally, in Section 7, we combine Lemmas 2, 3, and 4 to bound the maximum number of ancestors of blue vertices in its knowledge graph during the execution of any Q^*-query algorithm on $\boldsymbol{G} \sim \mathbb{BR}$. This is used to show that every such algorithm finds a cycle in $\boldsymbol{G} \sim \mathbb{BR}$ with probability $o_N(1)$.

To simplify the presentation, we introduce in Section 5.1 an augmented query model, called the *color revelation model*, in which more information is provided to the algorithm than in the standard model. Specifically, at the end of each epoch the query algorithm is provided with the color of every vertex it has previously seen. All our results discussed above are proved under this model and our lower bound trivially carries over to the standard model since any algorithm under the latter can be simulated under the color revelation model by simply ignoring the additional information.

4 The Bender-Ron graphs

In this section we formally describe the distribution $\mathbb{BR} := \mathbb{BR}(N, d)$ and prove, in Section 4.2, that $\boldsymbol{G} \sim \mathbb{BR}$ is 1/60-far from acyclic with probability $1 - o_N(1)$ when $d \geq 80$. Theorem 1 follows from the next theorem which we prove in the rest of the paper.

Theorem 2. *Let d be a constant with $d \geq 80$. Let $\mathcal{A}$ be any Q^*-query deterministic algorithm that operates on graphs in the support of $\mathbb{BR}$ under the vertex-query model, where $Q^* := N^{5/9}/\log(N)$. Then the probability of $\mathcal{A}$ finding a cycle in $\boldsymbol{G} \sim \mathbb{BR}$ is $o_N(1)$.*

4.1 The distribution.

Let $W := 2N/L = (2N)^{7/9}$ and $L := (2N)^{2/9}$ be two parameters indicating the width of each red layer and the number of red layers, respectively.[7] We refer to a map from a subset of $[3N]$ to $L+1$ colors $\{\text{blue}, \text{red}_1, \ldots, \text{red}_L\}$ as a *coloring*.

A digraph $\boldsymbol{G} \sim \mathbb{BR}$ over the vertex set $[3N]$ is generated by the following randomized procedure:

1. Let $\mathbb{U}$ be the uniform distribution over all colorings $C : [3N] \to \{\text{blue}, \text{red}_1, \ldots, \text{red}_L\}$ such that N vertices are colored blue and W vertices are colored red_i for each $i \in [L]$. The procedure starts by drawing a coloring $\boldsymbol{C} \sim \mathbb{U}$. Naturally we refer to vertices in $\mathbf{B}$ as blue vertices and vertices in $\mathbf{R}_1 \cup \cdots \cup \mathbf{R}_L$ as red vertices in $\boldsymbol{C}$. We view $\mathbf{R}_1, \ldots, \mathbf{R}_L$ as L layers of red vertices and refer to vertices in $\mathbf{R}_i$ as red vertices in the i^{th} layer (see Figure 1).
2. For each blue vertex $u \in \mathbf{B}$, create its adjacency list by drawing a sequence of d vertices *without* replacement from the following set of $(N-1) + LW/2 = 2N - 1$ vertices:
$$(\mathbf{B} \setminus \{u\}) \cup \bigcup_{i=1}^{L/2} \mathbf{R}_i.$$
Thus, a blue vertex has d distinct outneighbors from $\mathbf{B}$ and the top $L/2$ layers of red vertices.
3. For each red vertex in $\mathbf{R}_i$, $1 \leq i < L$, create its adjacency list by drawing a sequence of d vertices without replacement from $\mathbf{R}_{i+1}$. Thus, each red vertex (other than those in the bottom layer $\mathbf{R}_L$) has d distinct outneighbors in the next layer. Finally, set the adjacency list of each vertex in $\mathbf{R}_L$ to be empty. This finishes the construction of $\boldsymbol{G}$. Note that every vertex in $\boldsymbol{G}$ has out-degree either d or 0 so $\boldsymbol{G}$ is a bounded-outdegree-d digraph as promised.

We refer to graphs in the support of $\mathbb{BR}$ as Bender-Ron graphs, since these graphs are inspired by a construction that was proposed (but not analyzed) in [BR02]. Figure 1 illustrates a graph in $\mathbb{BR}$. To facilitate our proof of Theorem 2 later, in addition we introduce $\mathbb{BR}^*$ to denote the distribution of $(\boldsymbol{C}, \boldsymbol{G})$ generated by the procedure above (so the marginal distribution of $\boldsymbol{G}$ in $\mathbb{BR}^*$ is the same as $\mathbb{BR}$).

We record the following property that is trivial from the construction:

Property 1. *Let (C, G) be a pair in the support of $\mathbb{BR}^*$ and let (u, v) be an edge in G. Then either (1) $C(u) = C(v) = \text{blue}$ (a blue $\to$ blue edge), (2) $C(u) = \text{blue}$ and $C(v) = \text{red}_i$ for some $i \leq L/2$ (a blue $\to$ red edge), or (3) $C(u) = \text{red}_i$ and $C(v) = \text{red}_{i+1}$ for some $i < L$ (a red $\to$ red edge).*

Moreover, if a vertex u has no outneighbor, then we must have $C(u) = \text{red}_L$.

4.2 Almost all Bender-Ron graphs are far from acyclic.

It is clear from the construction that no red vertex can participate in a cycle, but intuitively the blue $\to$ blue edges will result in many cycles in the blue part of the graph. Lemma 1 below makes this intuition precise.

[7] Note that by our choices of L and W, $N + LW = 3N$. This particular setting of L and W is chosen to optimize our lower bound, as will become clear in the course of our analysis. We assume without loss of generality that N is such that both $L/2$ and W are integers.

Lemma 1 ($\mathbb{BR}$-graphs are far from acyclic)**.** *Let $d \geq 80$ be a constant. Then a random digraph $\boldsymbol{G} \sim \mathbb{BR}$ is $1/60$-far from acyclic with probability $1 - o_N(1)$.*

Proof. It suffices to show that for any fixed coloring C, the random graph $\boldsymbol{G}$ drawn using the same procedure running on C is far from acyclic with high probability. To this end, we assume without loss of generality that the blue vertices in C are $[N]$. We focus on the subgraph of $\boldsymbol{G}$ induced by the blue vertices $[N]$, which we refer to as the *blue subgraph*.

The following claim is folklore and we include its proof for completeness:

Claim 1. *An N-vertex digraph $G = (V, E)$ is ϵ-far from acyclic if and only if for every (bijective) vertex ordering $\pi : V \to \{1, \ldots, N\}$, the number of "backedges" (i.e. directed edges (u, v) such that $\pi(u) > \pi(v)$) is at least ϵdN.*

Proof. We prove the contrapositive in both directions: ($\Rightarrow$) Deleting all the backedges leaves an acyclic graph, showing that the graph is ϵ-close to acyclic. ($\Leftarrow$) Given a feedback arc set, after deleting it we can find a topological sort of the resulting acyclic graph. The ordering resulting from the topological sort has exactly the feedback arc set as its backedges. □

We will use the following claim, which follows trivially from Claim 1, to bound the distance to acyclicity of the blue subgraph of $\boldsymbol{G}$:

Claim 2. *Let $G = (V, E)$ be an N-vertex digraph. Suppose that for all balanced partitions (V_1, V_2) of V, the number of directed edges from V_1 to V_2 is at least ϵdN, then G is ϵ-far from acyclic.*

Proof. Every ordering of vertices π induces a balanced partition (V_1, V_2) by taking V_2 as the first $N/2$ vertices in π and V_1 as the last $N/2$ vertices in π. Then all edges from V_1 to V_2 are backedges with respect to π. The result follows from Claim 1. □

Fix a balanced partition (V_1, V_2) of the blue vertices $[N]$. We show below that the number of edges from V_1 to V_2 in $\boldsymbol{G}$ is at least $dN/20$ with probability $1 - \exp(-N)$. It follows from a union bound over all balanced partitions that with probability $1 - o_N(1)$, the number of edges one needs to delete to make $\boldsymbol{G}$ acyclic is at least $dN/20$.

To bound the number of edges in $\boldsymbol{G}$ from V_1 to V_2, we go through vertices in V_1 one by one and for each vertex $u \in V_1$, draw a sequence of d outneighbors without replacement from a set of $2N - 1$ vertices which contains V_2. For each of these $dN/2$ many rounds and for any outcomes in previous rounds, the probability of gaining a directed edge from V_1 to V_2 is at least

$$\frac{(N/2) - (d-1)}{2N - 1 - (d-1)} \geq \frac{1}{5}$$

when N is sufficiently large, so the expected number of edges is at least $(dN/2) \cdot (1/5) = dN/10$. It follows from a Chernoff bound (and a standard coupling argument) that the probability of having fewer than $dN/20$ edges is at most

$$e^{-(dN/10)(1/2)^2(1/2)} = e^{-dN/80} \leq e^{-N},$$

when $d \geq 80$. With a union bound over the at most 2^N many balanced partitions, we conclude that $\boldsymbol{G}$ has at least $dN/20$ edges from V_1 to V_2 in all balanced partitions (V_1, V_2) of $[N]$ with probability at least $1 - \exp(-N) \cdot 2^N = 1 - o_N(1)$. Thus the blue subgraph is $(1/20)$-far from acyclic with probability $1 - o_N(1)$. Because the total number of vertices is $3N$, after lifting back to the original graph we have that $\boldsymbol{G}$ is $(1/60)$-far from acyclic with probability $1 - o_N(1)$. □

5 Epochs and color revelation

The goal of the rest of the paper is to prove Theorem 2. Recall that under the vertex query model, each time an algorithm queries a vertex $u \in [3N]$ it receives as its answer an ordered list $a = (v_1, \ldots, v_\ell)$ containing the outneighbors of u. We assume without loss of generality that the algorithm never queries the same vertex twice. For Bender-Ron graphs in the support of $\mathbb{BR}$ we know that the answer to each query is either an ordered list $(v_1, \ldots, v_d)$ of d distinct vertices different from u or the empty list. This leads to the following definition of *query histories*.

Definition 1 (Query histories)**.** *A* query history H *is an ordered tuple $((u_1, a_1), \ldots, (u_q, a_q))$ for some $q \geq 0$ such that $u_1, \ldots, u_q$ are* distinct *vertices in $[3N]$ and each a_i is either a list of d distinct vertices different from u_i or the empty list. We refer to q as the length of H, and H as the empty history when $q = 0$.*

Each query history $H = ((u_1, a_1), \ldots, (u_q, a_q))$ uniquely determines a *knowledge graph*, denoted $\mathsf{KG}(H)$, which summarizes the information about the underlying graph contained in H: The vertex set of $\mathsf{KG}(H)$, denoted $\mathsf{VKG}(H)$, consists of all vertices that appear in H (i.e., every u_i and every vertex v in a_i for some $i \in [q]$); $\mathsf{KG}(H)$ contains a directed edge (u, v) if $u = u_i$ and v appears in a_i for some $i \in [q]$. Note that each vertex in $\mathsf{KG}(H)$ has outdegree either d or 0, and every vertex with outdegree d is queried in H. On the other hand, a vertex u with outdegree 0 has two cases: Either

u is queried in H and the answer a is empty, in which case we refer to u as a *sink* in $\mathsf{KG}(H)$, or u is discovered as an outneighbor of some vertex queried in H but itself is never queried in H.

To prove Theorem 2, we introduce the notion of *epochs* and a new query model called the *color revelation model* in Section 5.1. In addition to receiving the adjacency list of the vertex queried, an algorithm under the color revelation model receives additional information about colors of vertices in the current knowledge graph at the end of each epoch. In the rest of the paper we show that, under the color revelation model, any Q^*-query deterministic algorithm finds a cycle in $\boldsymbol{G} \sim \mathbb{BR}$ with probability $o_N(1)$ (see the exact statement in Theorem 3). Theorem 2 follows from Theorem 3 trivially because the color revelation model is no harder than the vertex query model: any algorithm under the vertex query model can be simulated under the color revelation model by simply ignoring the additional information.

5.1 The color revelation model. Let $H = ((u_1, a_1), \ldots, (u_q, a_q))$ be a query history for some $q \geq 0$; we write H_i to denote its i-prefix $((u_1, a_1), \ldots, (u_i, a_i))$. We say the k^{th} query (u_k, a_k) is a *surprise* in H if a_k contains a vertex that appears in $\mathsf{VKG}(H_{k-1})$. Otherwise, we refer to (u_k, a_k) as *surprise-free*.

We now describe the color revelation model, which provides additional power to the query algorithm by revealing the colors of vertices in previous epochs for free. Although this augmentation makes the task of cycle-finding easier, it also makes it easier to prove lower bounds. Formally, the oracle now contains a pair (C, G) in the support of $\mathbb{BR}^*$, instead of just a Bender-Ron graph G as in the vertex-query model. The oracle uses C to reveal to the algorithm colors of certain vertices. (In general, a coloring C is not uniquely determined by a Bender-Ron graph G.)

Under the color revelation model, an algorithm $\mathcal{A}$ maintains a triple $(H, \mathcal{E}, P)$, where

1. H is the current query history, updated after each query as in the vertex-query model;
2. $\mathcal{E} = (E_1, \ldots, E_\ell)$ for some $\ell \geq 1$ is a decomposition of H into *epochs*, where each epoch E_i is by itself a query history and $H = E_1 \circ \cdots \circ E_\ell$; and
3. Letting $H' = E_1 \circ \cdots \circ E_{\ell-1}$, P is a coloring map from $\mathsf{VKG}(H')$ to $\{\text{blue}, \text{red}_1, \ldots, \text{red}_L\}$.

Initially, H and E_1 are empty and $\mathcal{E} = (E_1)$. We refer to the final epoch E_l as the *current* epoch. For clarity, we use the symbols P and S to denote partial colorings over subsets of $[3N]$ and use C to denote a full coloring over the vertex set $[3N]$.

Let $(H, \mathcal{E}, P)$ denote the current triple maintained by an algorithm $\mathcal{A}$. Under the color revelation model, the next round proceeds as follows:

1. As in the vertex query model, $\mathcal{A}$ queries a vertex u, receives an ordered list a containing the outneighbors of u in G, and concatenates (u, a) to H and E_ℓ.
2. The current epoch *ends* if (u, a) is a surprise in H or $|E_\ell| = L/2$. In this case:
   (a) $\mathcal{A}$ learns the colors of the vertices in the current epoch: P is extended so that $P(u) = C(u)$ for every $u \in \mathsf{VKG}(H)$.
   (b) A new epoch begins: An empty epoch $E_{\ell+1}$ is appended to $\mathcal{E}$.

Note that $\mathcal{E}$ can be reconstructed from H by reading H serially and recording the end of an epoch if a surprise occurs or the length of the epoch reaches $L/2$. Thus $\mathcal{A}$ needs only to maintain the pair (H, P) instead of the triple $(H, \mathcal{E}, P)$. We refer to $\mathcal{E}$ as the *epoch decomposition* of H.

Next we introduce the notion of *valid knowledge pairs*.

Definition 2 (Valid knowledge pairs)**.** *A pair (H, P) is called a* valid knowledge pair *if*

- *$H = ((u_1, a_1), \ldots, (u_q, a_q))$ is a query history for some $q \geq 0$ and P is a coloring map over $\mathsf{VKG}(H')$, where $\mathcal{E} = (E_1, \ldots, E_\ell)$ is the epoch decomposition of H and $H' = E_1 \circ \cdots \circ E_{\ell-1}$;*
- *There exists a pair (C, G) in the support of $\mathbb{BR}^*$ such that C is an extension of P and G is* consistent *with H, i.e., a_i is the adjacency list of u_i in G for every $i \in [q]$.*

Given a valid knowledge pair (H, P) we use $\mathbb{BR}^*(H, P)$ to denote the distribution of $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$ conditioning on $\boldsymbol{C}$ being an extension of P and $\boldsymbol{G}$ being consistent with H.

Note that the pair (H, P) maintained by an algorithm under the color revelation model is always valid by definition. From now on we consider a deterministic query algorithm $\mathcal{A}$ under the color revelation model as a map from valid knowledge pairs to vertices so that $u = A(H, P)$ is the next vertex that is queried. Theorem 2 follows directly from the following statement in the color revelation model:

Theorem 3. *Let d be a constant with $d \geq 80$, and let $\mathcal{A}$ be a Q^*-query deterministic algorithm that works on pairs in the support of $\mathbb{BR}^*$ under the color revelation model, where $Q^* = N^{5/9}/\log N$. Then the probability of $\mathcal{A}$ finding a cycle in $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$ is $o_N(1)$.*

5.2 Epoch bounds. Let (H, P) be a valid knowledge pair and let $\mathcal{E} = (E_1, \ldots, E_\ell)$ be the epoch decomposition of the query history H. We refer to an epoch E_i, $i < \ell$, as a *surprise epoch* if its last query is a surprise in H; otherwise E_i has length $L/2$ and ends by timeout. A surprise epoch E_i is a *blue surprise epoch* if the last vertex queried in E_i is blue in P.

We begin our proof of Theorem 3 by proving upper bounds on the number of epochs and blue surprise epochs that occur during the execution of a Q-query algorithm under the color revelation model.

Lemma 2 (Epoch bound)**.** *Let $Ep(k)$ denote the event that more than k epochs occur. There exists a constant c_1 such that for any algorithm that makes Q queries,*

(5.1)
$$\Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[Ep\left(c_1\left(\frac{Q^2}{W}+\frac{Q}{L}\right)\right)\right] \le \exp\left(-\Omega\left(\frac{Q^2}{W}\right)\right).$$

Proof. Let $\mathcal{A}$ be an algorithm that makes Q queries. Since each epoch is either a surprise epoch or ends by timeout, the number of epochs which take place in running $\mathcal{A}$ on a pair (C, G) in the support of $\mathbb{BR}^*$ is bounded from above by the number of surprise queries plus $2Q/L$. As a result, it suffices to show that the probability of $\mathcal{A}$ observing more than $O(Q^2/W)$ many surprises when running on $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$ is at most $\exp(-\Omega(Q^2/W))$.

For this purpose we fix a valid knowledge pair (H, P) and let $u = \mathcal{A}(H, P)$ be the vertex that $\mathcal{A}$ queries next. Below we upper bound the probability of u being a surprise by $O(Q/W)$ when $\mathcal{A}$ runs on $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*(H, P)$. Since u has not been queried before, a key observation is that, fixing any coloring C in the support of $\mathbb{BR}^*(H, P)$ and conditioning $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*(H, P)$ further on $\boldsymbol{C} = C$, the adjacency list of u is distributed as follows: If $C(u) =$ blue then each of its d outneighbors is drawn without replacement from vertices of color blue in C (other than u itself) and vertices of color red_i, $i \le L/2$; If $C(u) = \mathrm{red}_i$ for some $i < L$ then each of its d outneighbors is drawn without replacement from vertices of color red_{i+1} in C; If $C(u) = \mathrm{red}_L$, then its adjacency list is empty.

As a result, if $C(u) =$ blue, the probability of u being a surprise query (as $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*(H, P)$ further conditioning on $\boldsymbol{C} = C$) is at most

$$d \cdot \frac{Q(d+1)}{2N-1} \le \frac{d^2 Q}{N},$$

using a union bound and the fact that $\mathsf{VKG}(H)$ has size at most $q(d+1) \le Q(d+1)$. Similarly the probability of u being a surprise when $C(u) = \mathrm{red}_i$ for some $i < L$ can be bounded from above by $2d^2Q/W$. Since u is always surprise-free if $C(u) = \mathrm{red}_L$, we have that the probability of u being a surprise when $\mathcal{A}$ runs on $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*(H, P)$ is at most $2d^2Q/W$.

Now for each $q \in [Q]$, let $\boldsymbol{X}_q$ be a Bernoulli random variable which is 1 if the q^{th} query made by $\mathcal{A}$ on $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$ is a surprise. Then what we have shown above implies that the probability of $\boldsymbol{X}_q = 1$ is $O(Q/W)$ even conditioning on any outcomes of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{q-1}$. It then follows from the Chernoff bound (together with a standard coupling argument) that

$$\Pr\left[\sum_{q\in[Q]} \boldsymbol{X}_q \ge \frac{4d^2Q^2}{W}\right] \le \exp\left(-\Omega\left(\frac{Q^2}{W}\right)\right).$$

This finishes the proof of the lemma. □

Recall that an epoch ends as a blue surprise epoch if the last query u is both a surprise and a blue vertex. If we let $\boldsymbol{X}_q$ denote the random variable that is 1 if the q^{th} query of $\mathcal{A}$ turns out to be the last query of a blue surprise epoch, when running on $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$, then the argument used in the proof of Lemma 2 implies that the probability of $\boldsymbol{X}_q = 1$ is at most $O(Q/N)$ conditioning on any outcomes of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{q-1}$. This gives us the following upper bound:

Lemma 3 (Blue surprise epochs bound)**.** *Let $BSEp(k)$ denote the event that more than k blue surprise epochs occur. There exists a constant c_2 such that for any algorithm that makes Q queries, we have*

(5.2)
$$\Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[BSEp\left(\frac{c_2Q^2}{N}\right)\right] \le \exp\left(-\Omega\left(\frac{Q^2}{N}\right)\right).$$

6 Bounding the probability of long blue paths

In this section we prove a key lemma necessary for the proof of Theorem 3: that in any given epoch, the probability that a Q^*-query algorithm discovers a "long" path of previously unseen blue vertices is low. As a result, with high probability, the subgraph induced by the blue nodes revealed at the end of each epoch is a forest in which every tree has small depth.

Lemma 4 (Long blue paths are unlikely.)**.** *Let (H, P) be a valid knowledge pair in which the length q of H is bounded by Q^*. Let E_ℓ be the current epoch of H and let $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*(H, P)$. The probability that $\mathsf{KG}(E_\ell)$ contains a path of length at least $4 \log N$ consisting of blue vertices only under $\boldsymbol{C}$ is $o(N^{-2})$.*[8]

We begin with some notation and a sketch of the proof. Let (H, P) be a valid knowledge pair, let $\mathcal{E} =$

[8] The constant factors in the lemma statement are arbitrary but will be convenient later in the analysis.

$(E_1, \ldots, E_\ell)$ be the epoch decomposition of H, and let $H' = E_1 \circ \cdots \circ E_{\ell-1}$. By definition, the current epoch E_ℓ satisfies $|E_\ell| < L/2$ and every query in E_ℓ is surprise-free in H. As a result, the graph $\mathsf{KG}(E_\ell)$ is a vertex-disjoint union of degree-d out-trees.[9] This leads to the following observation:

Property 2. *Let T be an out-tree in $\mathsf{KG}(E_\ell)$.*

1. *Every vertex of T, other than the root, lies outside $\mathsf{VKG}(H')$. (The root may or may not lie in $\mathsf{VKG}(H')$.)*
2. *Every internal vertex and every sink in T is queried in E_ℓ.*

Let $\boldsymbol{S} \sim \mathbb{S}$ be the distribution of partial colorings over $\mathsf{VKG}(H)$ induced by $\boldsymbol{C}$ drawn as in $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*(H, P)$. Then every partial coloring S in the support of $\mathbb{S}$ must be a *good* partial coloring over $\mathsf{VKG}(H)$ (see Property 1):

Definition 3. *We say S is a* good *partial coloring over $\mathsf{VKG}(H)$ if (1) S is an extension of P, (2) For each directed edge (u, v) in $\mathsf{KG}(H)$, either $S(u) = S(v) =$ blue, or $S(u) =$ blue and $S(v) = \text{red}_i$ for some $i \in [L/2]$, or $S(u) = \text{red}_i$ and $S(v) = \text{red}_{i+1}$ for some $i < L$, and (3) $S(u) = \text{red}_L$ for every sink vertex in H.*

In other words, Lemma 4 states that $\mathsf{KG}(E_\ell)$ is unlikely to have a long blue path under $\boldsymbol{S} \sim \mathbb{S}$. To prove Lemma 4, we introduce a *naive distribution* $\mathbb{S}'$ in Section 6.1 that is much easier to work with and at the same time serves as a good approximation of the distribution $\mathbb{S}$. We then show that $\mathsf{KG}(E_\ell)$ is unlikely to have a long blue path under $\boldsymbol{S}' \sim \mathbb{S}'$, from which Lemma 4 follows.

The intuition behind the naive distribution $\mathbb{S}'$ is that we color each tree T in $\mathsf{KG}(E_\ell)$ independently, ignoring all information in the knowledge pair (H, P) other than the tree T itself. Roughly speaking, we generate a coloring for T as follows. If the root of T lies outside of $\mathsf{VKG}(H')$, we color it red with probability $2/3$ and blue with probability $1/3$ as if it were drawn uniformly at random from $[3N]$. If the root of T lies inside $\mathsf{VKG}(H')$, its color is known. We then propagate down the tree in breadth-first order. If the parent of a vertex v was colored blue, v is colored blue with probability $1/2$ and red_i with probability $1/L$ for each $i \in [L/2]$; if the parent of v was colored red_i then v is colored red_{i+1}. $\mathbb{S}'$ does not capture $\mathbb{S}$ perfectly, but we show in Section 6.2 that they are pointwise very close to each other.

[9] Note that an isolated vertex is also counted as a degree-d out-tree.

6.1 The naive distribution.

Before introducing the naive distribution $\mathbb{S}'$, we start by classifying trees of $\mathsf{KG}(E_\ell)$ into four types and note that we can already deduce colors of certain vertices in any good coloring S over $\mathsf{VKG}(H)$. Let T be an out-tree of $\mathsf{KG}(E_\ell)$ with height $\mathsf{h}(T)$ and root vertex r:

- T is a *type*-1 out-tree if $r \in \mathsf{VKG}(H')$ (so the color of r has already been revealed in P) and $P(r) = \text{red}_i$ for some $i \in [L]$. It follows from Property 1 that every valid coloring S has $S(v) = \text{red}_{i+\ell}$ for each vertex v of depth ℓ in the tree. (Note that we must have $i + \mathsf{h}(T) \le L$; otherwise the pair (H, P) cannot be a valid knowledge pair.)
- T is a *type*-2 out-tree if $r \in \mathsf{VKG}(H')$ and $P(r) =$ blue. Then *none* of its leaves can be a sink; otherwise (H, P) implies that there is a path from a blue vertex to a red_L vertex of length at most $\mathsf{h}(T) \le |E_\ell| < L/2$, contradicting with the validity of (H, P).
- T is a *type*-3 out-tree if r is not in $\mathsf{VKG}(H')$ but T contains at least one sink leaf v^*. Given that $\mathsf{h}(T) < L/2$, it follows from Property 1 that every good coloring S satisfies $S(r) = \text{red}_{L-k}$, where k is the depth of v^* in T, and $S(v) = \text{red}_{L-k+\ell}$ for every vertex v of depth ℓ in the tree.
- T is a *type*-4 out-tree if r is not in $\mathsf{VKG}(H')$ and none of its leaves is a sink.

Figure 2 illustrates the four types of out-trees. Let U denote the set of vertices that are always colored red or always colored blue in a good partial coloring for $\mathsf{VKG}(H)$; that is, every vertex in $\mathsf{VKG}(H')$ as well as vertices in type-1 and type-3 trees in $\mathsf{KG}(E_l)$. Let P' denote the unique partial coloring over U that agrees with every good partial coloring S over $\mathsf{VKG}(H)$. We let $Y := \mathsf{VKG}(H) \setminus U$ denote the set of vertices which may be colored either red or blue in a good coloring; that is, all vertices in type-2 and type-4 trees except for the roots of type-2 trees.

We are ready to define the *naive distribution* $\mathbb{S}'$ of partial colorings over $\mathsf{VKG}(H)$. A coloring $\boldsymbol{S}' \sim \mathbb{S}'$ is drawn using the following procedure:

1. First we color each vertex $u \in U$ as $P'(u)$ (so $\boldsymbol{S}'$ is always an extension of P');
2. For each type-4 tree T, color its root vertex r blue with probability $N/(3N - \mathsf{h}(T)W)$ and red_i with probability $Y/(3N - \mathsf{h}(T)W)$ for each $i \le L - \mathsf{h}(T)$. (The intuition behind the denominator is that because there is a path of

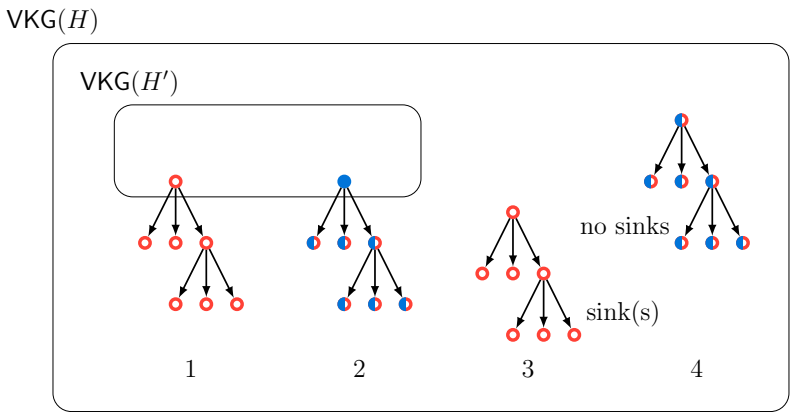

Figure 2: Possible out-tree types during a given epoch. Bichromatic circles denote vertices whose types cannot be determined from (H, P) alone.

length $\mathsf{h}(T)$ that starts at r, its color cannot be $\mathrm{red}_{L-\mathsf{h}(T)+1}, \ldots, \mathrm{red}_L$.)

3. Then we go through each type-2 and type-4 tree one by one and consider uncolored vertices in breadth-first order. For each vertex v, if its parent is colored blue, color v blue with probability $1/2$ and with red_i for each $i \in [L/2]$ with probability $1/L$. If the parent of v is colored red_i, color v red_{i+1}.[10]

The following property follows directly from the procedure for $\mathbb{S}'$ above:

Property 3. *Every partial coloring in the support of* $\mathbb{S}'$ *is a good partial coloring over* $\mathsf{VKG}(H)$.

Both distributions $\mathbb{S}$ and $\mathbb{S}'$ are supported on good partial colorings over $\mathsf{VKG}(H)$. The next lemma shows that $\mathbb{S}'$ is a good approximation of $\mathbb{S}$:

Lemma 5. *For every good partial coloring* S *over* $\mathsf{VKG}(H)$*, we have*

$$0.9 \cdot \Pr_{\boldsymbol{S}' \sim \mathbb{S}'}\left[\boldsymbol{S}' = S\right] \le \Pr_{\boldsymbol{S} \sim \mathbb{S}}\left[\boldsymbol{S} = S\right] \le 1.1 \cdot \Pr_{\boldsymbol{S}' \sim \mathbb{S}'}\left[\boldsymbol{S}' = S\right].$$

Before proving Lemma 5 in Section 6.2, we use it to give a quick proof of Lemma 4.

Proof of Lemma 4 using Lemma 5. Given a good coloring S over $\mathsf{VKG}(H)$ we use $\mathrm{LBP}(S)$ to denote the event that $\mathsf{KG}(E_\ell)$ contains a blue path of length at least $4 \log N$ under S. It follows from Lemma 5 that

$$\Pr_{\boldsymbol{S} \sim \mathbb{S}}\left[\mathrm{LBP}(\boldsymbol{S})\right] \le 1.1 \cdot \Pr_{\boldsymbol{S}' \sim \mathbb{S}'}\left[\mathrm{LBP}(\boldsymbol{S}')\right]. \tag{6.3}$$

[10] Observe that we never go beyond red_L because the height of each tree is at most $L/2$.

On the other hand, if $\mathrm{LBP}(S)$ holds then there must be a vertex v in either a type-2 or a type-4 tree (since every vertex in a type-1 or type-3 tree must be red in a good partial coloring) such that v is of depth at least $4 \log N$ and the path from the root to v is all blue. For each such vertex v, let $\mathrm{LBP}(S, v)$ denote the event that the path from the root to v is blue under S. Then the probability of $\mathrm{LBP}(\boldsymbol{S}', v)$ when $\boldsymbol{S}' \sim \mathbb{S}'$ is $(1/2)^\ell \le 1/N^4$ if v is in a type-2 tree and has depth ℓ, and is

$$\frac{N}{3N - \mathsf{h}(T)W} \cdot (1/2)^\ell \le 1/N^4$$

if v is in a type-4 tree T. As a result, the probability of $\mathrm{LBP}(\boldsymbol{S}')$ when $\boldsymbol{S}' \sim \mathbb{S}'$ is $O(1/N^3)$ by a union bound since the number of v is trivially at most $3N$. The lemma follows from (6.3). □

6.2 The naive distribution is a good approximation: Proof of Lemma 5.

To simplify the presentation, in this section we use the notation "$a \pm b$" to denote a quantity that is between $a - b$ and $a + b$.

Let S be a good partial coloring over $\mathsf{VKG}(H)$. We write $\mathcal{T}_2$ to denote the set of type-2 trees in $\mathsf{KG}(E_\ell)$ and $\mathcal{T}_4$ to denote the set of type-4 trees in $\mathsf{KG}(E_\ell)$, and we write $\mathcal{T}$ to denote $\mathcal{T}_2 \cup \mathcal{T}_4$. Given S, we write $\mathcal{T}_{4,b}(S)$ to denote the set of type-4 trees with a blue root and $\mathcal{T}_{4,r}(S)$ to denote the set of type-4 trees with a red root in S. We also use $\#_{br}(S), \#_{bb}(S)$ and $\#_{rr}(S)$ to denote the total number of blue $\to$ red, blue $\to$ blue and red $\to$ red edges in all trees in $\mathcal{T}$.

We start with the easier task of obtaining a closed-form expression for $\Pr_{\boldsymbol{S}' \sim \mathbb{S}'}[\boldsymbol{S}' = S]$. This quantity can be written as a product: each root of a type-4 tree contributes a factor which depends on its color in S (recall the second step of the procedure for drawing from $\mathbb{S}'$), and each edge of a tree in $\mathcal{T}$ contributes a

factor which is $1/2$ if it is a blue $\to$ blue edge in S, $1/L$ if it is a blue $\to$ red edge, and 1 if it is a red $\to$ red edge. As a result, we have

$$\begin{aligned}
&\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}\left[\boldsymbol{S}'=S\right]\\
&=\left(\prod_{T\in\mathcal{T}_{4,b}(S)}\frac{N}{3N-\mathsf{h}(T)W}\right)\left(\prod_{T\in\mathcal{T}_{4,r}(S)}\frac{W}{3N-\mathsf{h}(T)W}\right)\\
&\qquad\left(\frac{1}{L}\right)^{\#_{br}(S)}\left(\frac{1}{2}\right)^{\#_{bb}(S)}\\
&=\left(\prod_{T\in\mathcal{T}_4}\frac{W}{3N-\mathsf{h}(T)W}\right)\left(\frac{L}{2}\right)^{|\mathcal{T}_{4,b}(S)|}\left(\frac{1}{L}\right)^{\#_{br}(S)}\\
&\qquad\left(\frac{1}{2}\right)^{\#_{bb}(S)}\\
&=\tau_1\cdot\left(\frac{L}{2}\right)^{|\mathcal{T}_{4,b}(S)|}\left(\frac{1}{L}\right)^{\#_{br}(S)}\left(\frac{1}{2}\right)^{\#_{bb}(S)},
\end{aligned}$$

where the second equality uses $WL/2=N$ and the fact that $\mathcal{T}_4$ is the disjoint union of $\mathcal{T}_{4,b}(S)$ and $\mathcal{T}_{4,r}(S)$. The quantity $\tau_1>0$ is a value that does not depend on S.

Next we work on the probability distribution $\mathbb{S}$. For each good partial coloring S over $\mathsf{VKG}(H)$, we write $\mathsf{w}(S)$ as a shorthand for the probability over $(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*$ that $\boldsymbol{C}$ is an extension of S and $\boldsymbol{G}$ is consistent with H. Given the definition of $\mathbb{S}$ and $\mathsf{w}(\cdot)$, we have

$$\Pr_{\boldsymbol{S}\sim\mathbb{S}}\left[\boldsymbol{S}=S\right]=\frac{\mathsf{w}(S)}{\sum_{\text{good }S'}\mathsf{w}(S')},\tag{6.4}$$

where the sum is over all good partial colorings S' over $\mathsf{VKG}(H)$.

Looking ahead, our plan is to show that there is a value $\tau>0$ (independent of $\boldsymbol{S}$) such that

$$0.99\tau\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}\left[\boldsymbol{S}'=S\right]\le\mathsf{w}(S)\le1.01\tau\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}\left[\boldsymbol{S}'=S\right],$$

or more succinctly,

$$\mathsf{w}(S)\in(1\pm0.01)\tau\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}\left[\boldsymbol{S}'=S\right].\tag{6.5}$$

With the above expression, it follows from

$$\sum_{\text{good }S'}\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}[\boldsymbol{S}'=S']=1$$

(since $\mathbb{S}'$ is supported on good colorings) that

$$0.99\tau\le\sum_{\text{good }S'}\mathsf{w}(S')\le1.01\tau.\tag{6.6}$$

Combining (6.4), (6.5), and (6.6), we have

$$\Pr_{\boldsymbol{S}\sim\mathbb{S}}\left[\boldsymbol{S}=S\right]\le\frac{1.01\tau}{0.99\tau}\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}\left[\boldsymbol{S}'=S\right]<1.1\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}\left[\boldsymbol{S}'=S\right]$$

and the other side of Lemma 5 can be proved similarly. Thus, it suffices to prove (6.5). We start with some notation. We use $\mathbb{U}$ to denote the uniform distribution over all full colorings. Given a full coloring C, we use $\mathbb{BR}(C)$ to denote the distribution of Bender-Ron graphs generated using C as the full coloring in the procedure for $\mathbb{BR}$.

Now we consider the $(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*$ in the definition of $\mathsf{w}(S)$ by first drawing a full coloring $\boldsymbol{C}\sim\mathbb{U}$. If $\boldsymbol{C}$ is not an extension of S then we already fail to satisfy the condition in the definition of $\mathsf{w}(S)$. Let $\mathrm{ext}(\boldsymbol{C},S)$ denote the event that $\boldsymbol{C}$ is an extension of S. If $\mathrm{ext}(\boldsymbol{C},S)$ then we draw $\boldsymbol{G}\sim\mathbb{BR}(\boldsymbol{C})$ to see if $\boldsymbol{G}$ is consistent with H.

A useful observation is that every C that extends S shares the same probability of $\boldsymbol{G}\sim\mathbb{BR}(C)$ being consistent with H. Let $\#_b(U)$ (respectively $\#_r(U)$) be the number of blue (respectively red) vertices in U under S that are *queried* in H; note that these two numbers are independent of S since every good coloring must be an extension of P' on U. Let $\#_b(Y,S)$ (respectively $\#_r(Y,S)$) denote the number of blue (respectively red) vertices in Y under S that are *queried* in H. Then for every C that is an extension of S, the probability of $\boldsymbol{G}\sim\mathbb{BR}(C)$ being consistent with H is

$$\begin{aligned}
&\left(\prod_{i=1}^{d}\frac{1}{2N-i}\right)^{\#_b(U)+\#_b(Y,S)}\left(\prod_{i=0}^{d-1}\frac{1}{W-i}\right)^{\#_r(U)+\#_r(Y,S)}\\
&=\tau_2\cdot\left(\prod_{i=1}^{d}\frac{1}{2N-i}\right)^{\#_b(Y,S)}\left(\prod_{i=0}^{d-1}\frac{1}{W-i}\right)^{\#_r(Y,S)}
\end{aligned}\tag{6.7}$$

for some positive value τ_2 independent of S. Note that our choices of L,W and Q^* satisfy

$$LQ^*=o(W).\tag{6.8}$$

Using (6.8) (we only need $L=o(W)$ here) and the fact that $\#_b(Y,S),\#_r(Y,S)\le L/2$, (6.7) becomes

$$\begin{aligned}
&=(1\pm o_N(1))\tau_2\left(\frac{1}{2N}\right)^{d\#_b(Y,S)}\left(\frac{1}{W}\right)^{d\#_r(Y,S)}\\
&=(1\pm o_N(1))\tau_3\left(\frac{1}{L}\right)^{d\#_b(Y,S)},
\end{aligned}$$

for some positive value τ_3 that is independent of S since $\#_b(Y,S)+\#_r(Y,S)$ is a constant independent of S.

Note that $d \cdot \#_b(Y,S) = \#_{bb}(S) + \#_{br}(S) - d|\mathcal{T}_2|$. This is just because each blue vertex queried in Y introduces d edges that are either blue $\to$ blue or blue $\to$ red in $\mathcal{T}$; we need to subtract $d|\mathcal{T}_2|$ because roots of type-2 trees are not included in Y. Since $|\mathcal{T}_2|$ is a value independent of S, (6.7) can be simplified to

$$(1 \pm o_N(1)) \cdot \tau_4 \cdot \left(\frac{1}{L}\right)^{\#_{bb}(S)+\#_{br}(S)},$$

for some value $\tau_4 > 0$ that is independent of S, and thus

$$\mathsf{w}(S) = (1 \pm o_N(1)) \Pr_{\boldsymbol{C}\sim\mathbb{U}}\left[\mathrm{ext}(\boldsymbol{C},S)\right]\tau_4 \left(\frac{1}{L}\right)^{\#_{bb}(S)+\#_{br}(S)}.$$

Next, we evaluate the probability that $\boldsymbol{C} \sim \mathbb{U}$ is an extension of S over $\mathsf{VKG}(H) = U \cup Y$. For this purpose we consider the following experiment:

1. Pick an arbitrary ordering $u_1, \dots, u_{|U|}$ of U and an arbitrary ordering $y_1, \dots, y_{|Y|}$ of Y.
2. Start with N blue pebbles and W red$_i$ pebbles for each $i \in [L]$. Go through vertices $u_1, \dots, u_{|U|}$ one by one and assign each one a remaining (as yet unassigned) pebble uniformly at random. Then go through vertices $y_1, \dots, y_{|Y|}$ one by one and assign each one a remaining pebble uniformly at random.
3. For each u_i, we use $\boldsymbol{X}_i$ to denote the Bernoulli random variable that is 1 if u_i is assigned a pebble of color $S(u_i)$, and define $\boldsymbol{Y}_i$ similarly for each y_i.

Then the probability $\Pr_{\boldsymbol{C}\sim\mathbb{U}}[\mathrm{ext}(\boldsymbol{C},S)]$ that we are interested in is

$$\begin{aligned}
&\Pr\left[\boldsymbol{X}_1 = \cdots = \boldsymbol{Y}_{|Y|} = 1\right] \\
&= \Pr\left[\boldsymbol{X}_1 = \cdots = \boldsymbol{X}_{|U|} = 1\right] \\
&\qquad \prod_{i\in[|Y|]} \Pr\left[\boldsymbol{Y}_i = 1 \,|\, \boldsymbol{X}_1 = \cdots = \boldsymbol{Y}_{i-1} = 1\right] \\
&= \tau_5 \cdot \prod_{i\in[|Y|]} \Pr\left[\boldsymbol{Y}_i = 1 \,|\, \boldsymbol{X}_1 = \cdots = \boldsymbol{Y}_{i-1} = 1\right],
\end{aligned}$$

for some positive value τ_5 that is independent of S. For each y_i with $S(y_i) =$ blue, we have

$$\frac{N - Q^*(d+1)}{3N} \le \Pr\left[\boldsymbol{Y}_i = 1 \,|\, \boldsymbol{X}_1 = \cdots = \boldsymbol{Y}_{i-1} = 1\right] \le \frac{N}{3N - Q^*(d+1)}.$$

This is because regardless of outcomes for vertices before y_i, the number of blue pebbles left in the round of y_i lies between $N - Q^*(d+1)$ and N (since $\mathsf{VKG}(H)$ has no more than $Q^*(d+1)$ vertices) and the total number of pebbles left is between $3N - Q^*(d+1)$ and $3N$.

Similarly for each y_i with $S(y_i) = \mathrm{red}_j$ for some $j \in [L]$, we have

$$\frac{W - Q^*(d+1)}{3N} \le \Pr\left[\boldsymbol{Y}_i = 1 \,|\, \boldsymbol{X}_1 = \cdots = \boldsymbol{Y}_{i-1} = 1\right] \le \frac{W}{3N - Q^*(d+1)}.$$

Let $\#_b^*(Y,S)$ (or $\#_r^*(Y,S)$) denote the number of blue (or red) vertices in Y under S (unlike $\#_b(Y,S)$ and $\#_r(Y,S)$, these vertices may have not been queried). It follows from (6.8) and $|Y| = O(L)$ that

$$\begin{aligned}
&\Pr_{\boldsymbol{C}\sim\mathbb{U}}\left[\mathrm{ext}(\boldsymbol{C},S)\right] \\
&= (1 \pm o_N(1))\,\tau_5 \left(\frac{1}{3}\right)^{\#_b^*(Y,S)} \left(\frac{W}{3N}\right)^{\#_r^*(Y,S)} \\
&= (1 \pm o_N(1))\,\tau_6 \left(\frac{2}{L}\right)^{\#_r^*(Y,S)},
\end{aligned}$$

for some value $\tau_6 > 0$ that is independent of S since $\#_b^*(Y,S) + \#_r^*(Y,S)$ is a constant independent of S. Finally we have

$$\begin{aligned}
&\frac{\mathsf{w}(S)}{\Pr_{\boldsymbol{S}'\sim\mathbb{S}'}[\boldsymbol{S}' = S]} \\
&= (1 \pm o_N(1))\,\tau_7 \left(\frac{2}{L}\right)^{\#_r^*(Y,S)+\#_{bb}(S)+|\mathcal{T}_{4,b}(S)|}
\end{aligned}$$

for some value $\tau_7 > 0$ that is independent of S. Note that for any good coloring S, the quantity $\#_r^*(Y,S) + \#_{bb}(S) + |\mathcal{T}_{4,b}(S)|$ is equal to $|Y|$, a constant that does not depend on S. This finishes the proof of (6.5) and Lemma 5.

7 A Lower Bound on Cycle Finding

This section combines the results of Lemmas 2, 3 and 4 to establish Theorem 3, which is restated below:

Theorem 3. *Let d be a constant with $d \ge 80$, and let $\mathcal{A}$ be a Q^*-query deterministic algorithm that works on pairs in the support of $\mathbb{BR}^*$ under the color revelation model, where $Q^* = N^{5/9}/\log N$. Then the probability of $\mathcal{A}$ finding a cycle in $(\boldsymbol{C},\boldsymbol{G}) \sim \mathbb{BR}^*$ is $o_N(1)$.*

Proof. Let $\mathcal{A}$ be a Q^*-query algorithm. We start with the definition of *typical* pairs in the support of $\mathbb{BR}^*$ with respect to $\mathcal{A}$, and then show that $(\boldsymbol{C},\boldsymbol{G}) \sim \mathbb{BR}^*$ is typical with probability $1 - o_N(1)$.

Definition 4. *We say a pair (C,G) in the support of $\mathbb{BR}^*$ is* typical *with respect to an algorithm $\mathcal{A}$ if the following conditions hold:*

(i) The number of epochs during the execution of $\mathcal{A}$ on (C, G) is $O(Q^{*2}/W + Q^*/L)$.

(ii) The number of blue surprise epochs during the execution of $\mathcal{A}$ on (C, G) is $O(Q^{*2}/N)$.

(iii) For each $q \in [Q^*]$, let $(H^{(q)}, P^{(q)})$ denote the knowledge pair of running $\mathcal{A}$ on (C, G) after q queries and let $E^{(q)}$ denote the current epoch (in the epoch decomposition of $H^{(q)}$). Then there is no blue path longer than $4 \log N$ in $\mathsf{KG}(E^{(q)})$ under the coloring C.

We combine Lemmas 2, 3 and 4 to show that $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$ is typical with respect to $\mathcal{A}$ with probability $1 - o_N(1)$. We focus on the third condition (iii) since the probability of $(\boldsymbol{C}, \boldsymbol{G})$ satisfying the first two conditions is $1 - o_N(1)$ by Lemmas 2 and 3. For (iii) we have

$$\Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[(\boldsymbol{C}, \boldsymbol{G}) \text{ violates (iii)}\right]$$
$$\leq \sum_{q=1}^{Q^*} \Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[(\boldsymbol{C}, \boldsymbol{G}) \text{ violates (iii) after } q \text{ queries}\right].$$

On the other hand, letting $\mathcal{A}_q(\boldsymbol{C}, \boldsymbol{G})$ denote the knowledge pair $\mathcal{A}$ observes after q queries on $(\boldsymbol{C}, \boldsymbol{G})$, the q^{th} probability in the sum can be written as

$$\sum_{\text{valid } (H,P)} \Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[\mathcal{A}_q(\boldsymbol{C}, \boldsymbol{G}) = (H, P)\right] \times$$
$$\Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*(H,P)}\left[(\boldsymbol{C}, \boldsymbol{G}) \text{ violates (iii) after } q \text{ queries}\right],$$

where the sum is over all valid knowledge pairs (H, P) of length q. It follows from Lemma 4 that the latter probability in the above expression is $o(N^{-2})$ for every valid knowledge pair (H, P). As a result, the probability of $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$ violating (iii) is $o(N^{-1})$ and thus $(\boldsymbol{C}, \boldsymbol{G})$ is typical with probability $1 - o_N(1)$.

Given a query history H and a vertex u, we write $\mathsf{anc}(H, u)$ to denote the set of ancestors of u in $\mathsf{KG}(H)$, i.e., the set of vertices (other than u itself) that have a directed path to u. (If $u \notin \mathsf{VKG}(H)$ then $\mathsf{anc}(H, u)$ is trivially empty.) The claim below shows that if (C, G) is typical then at any time during the execution of $\mathcal{A}$ on (C, G), every blue vertex has a *small* set of ancestors in $\mathsf{KG}(H)$.

Claim 3. *Let (C, G) be a typical pair with respect to $\mathcal{A}$. Then for each $q \in [Q^*]$, letting H be the query history of $\mathcal{A}$ after making q queries on (C, G), we have*

$$|\mathsf{anc}(H, u)| \leq O\left(\log N \cdot \left(\frac{Q^{*2}}{W} + \frac{Q^*}{L}\right) \cdot \frac{Q^{*2}}{N}\right), \tag{7.9}$$

for every vertex u with $C(u) = \text{blue}$.

Proof. Recall that at the end of each blue surprise epoch, $\mathcal{A}$ may find an edge (u, v) such that the vertex u being queried is blue and v is a vertex encountered before. We refer to such an edge as a *surprise edge* if v also turns out to be blue.

Now we consider running $\mathcal{A}$ on a typical pair (C, G). Let $(H^{(i)}, P^{(i)})$ denote the knowledge pair maintained by $\mathcal{A}$ after i queries, let $E^{(i)}$ be the current epoch, and let $H^{(i)} = H'^{(i)} \circ E^{(i)}$. We focus on the evolution of the blue subgraph (the subgraph induced by its blue vertices) of $\mathsf{KG}(H'^{(i)})$ over time. We write $\mathsf{BKG}(H'^{(i)})$ to denote the blue subgraph of $\mathsf{KG}(H'^{(i)})$.

First we note that $\mathsf{KG}(H'^{(i)})$ (and thus, $\mathsf{BKG}(H'^{(i)})$) is only updated at the end of each epoch. If an epoch ends at the i^{th} query, a number of out-trees are added to $\mathsf{KG}(H'^{(i-1)})$. Each such tree (other than its root) is vertex-disjoint from $\mathsf{KG}(H'^{(i-1)})$. In addition, if the epoch is a blue surprise epoch, no more than d many surprise edges are added to $\mathsf{KG}(H'^{(i)})$. Now focusing on $\mathsf{BKG}(H'^{(i)})$ vs $\mathsf{BKG}(H'^{(i-1)})$, we have that at the end of each epoch, each out-tree added to $\mathsf{BKG}(H'^{(i-1)})$ satisfies the extra condition of having depth at most $4 \log N$. If it is the end of a blue surprise epoch we may need to add no more than d surprise edges to $\mathsf{BKG}(H'^{(i)})$.

As a result, letting H be the query history of $\mathcal{A}$ after making q queries on a typical pair (C, G), we have that $\mathsf{BKG}(H')$ is the union of (1) a forest in which each out-tree has depth at most

$$O\left(\log N \cdot \left(\frac{Q^{*2}}{W} + \frac{Q^*}{L}\right)\right) \tag{7.10}$$

and (2) a set of at most

$$O\left(\frac{Q^{*2}}{N}\right) \tag{7.11}$$

many surprise edges, where (7.10) follows from the bound for the number of epochs and (7.11) follows from the bound for the number of blue surprise epochs given that (C, G) is typical.

Let u be a vertex in $\mathsf{BKG}(H')$. To bound the number of its ancestors, we consider an in-tree T rooted at u such that every ancestor of u appears in T (with a directed path to u). If we remove surprise edges from T, it is left with a vertex-disjoint union of directed paths; this is because, after removing surprise edges, $\mathsf{BKG}(H')$ is a forest of out-trees (so no vertex has indegree larger than 1). Since each path has length bounded by (7.10) and the number of surprise edges is bounded by (7.11), the number of vertices in T, i.e., the number of ancestors of u, is bounded by (7.9).

Now in Claim 3, u can be a vertex in $\mathsf{VKG}(E)$. Note that $\mathsf{KG}(E)$ must be a forest of out-trees and

because (C, G) is typical, the blue subgraph of each such out-tree has depth at most $4 \log N$. As a result, considering vertices in $\mathsf{VKG}(E)$ may add a term of $4 \log N$ to our bound for the number of ancestors, which is still captured by (7.9). This finishes the proof of the claim. □

Now we show that $\mathcal{A}$ finds a cycle in $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*$ with probability $o_N(1)$. Given that $(\boldsymbol{C}, \boldsymbol{G})$ is typical with probability at least $1 - o_N(1)$, and letting $cyc(\mathcal{A}_q, \boldsymbol{C}, \boldsymbol{G})$ denote the event that $\mathcal{A}$ finds a cycle on the qth query, we have

(7.12)
$$\begin{aligned}
&\Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[\mathcal{A} \text{ finds a cycle}\right] \\
&\le o_N(1) + \Pr_{(\boldsymbol{C},\boldsymbol{G})}\left[(\boldsymbol{C}, \boldsymbol{G}) \text{ typical}, \mathcal{A} \text{ finds a cycle}\right] \\
&\le o_N(1) + \sum_{q\in[Q^*]} \Pr_{(\boldsymbol{C},\boldsymbol{G})}\left[(\boldsymbol{C}, \boldsymbol{G}) \text{ typical}, cyc(\mathcal{A}_q, \boldsymbol{C}, \boldsymbol{G})\right].
\end{aligned}$$

As a result, it suffices to bound each probability in the sum by $o(1/N)$.

Fix any $q \in [Q^*]$. Given a pair (H, C), where $H = H' \circ E$ is a query history of length $q-1$ and C is a full coloring, we write $P(H, C)$ to denote the restriction of C on $\mathsf{VKG}(H')$. Then we have

$$\Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[(\boldsymbol{C}, \boldsymbol{G}) \text{ typical}, cyc(\mathcal{A}_q, \boldsymbol{C}, \boldsymbol{G})\right], \tag{7.13}$$

which is at most

(7.14)
$$\begin{aligned}
\sum_{(H,C)} &\Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*}\left[\boldsymbol{C} = C \text{ and } \mathcal{A} \text{ sees } (H, P(H, C))\right] \\
&\times \Pr_{(\boldsymbol{C},\boldsymbol{G})\sim\mathbb{BR}^*(H,C)}\left[cyc(\mathcal{A}_q, \boldsymbol{C}, \boldsymbol{G})\right],
\end{aligned}$$

where the sum is over all (H, C) such that every blue vertex (under C) in $\mathsf{VKG}(H)$ has its number of ancestors bounded by (7.9). This follows from Claim 3 since $(\boldsymbol{C}, \boldsymbol{G})$ is typical in (7.13).

Fix such a pair (H, C) and let $u = \mathcal{A}(H, P(H, C))$ be the next vertex that is queried by $\mathcal{A}$. If $u \notin \mathsf{VKG}(H)$ or $u \in \mathsf{VKG}(H)$ is not blue under C, the probability in (7.14) is 0. If $u \in \mathsf{VKG}(H)$ is blue, we note that under $(\boldsymbol{C}, \boldsymbol{G}) \sim \mathbb{BR}^*(H, C)$, outneighbors of u are picked randomly from $2N - 1$ vertices without replacement. Consequently the probability that one of them is an ancestor of u can be bounded by

$$O\left(\log N \cdot \left(\frac{Q^{*4}}{WN^2} + \frac{Q^{*3}}{LN^2}\right)\right).$$

As a result, this is an upper bound for (7.14) as well as (7.13) and thus, the sum in (7.12) is at most

$$\begin{aligned}
&Q^* \cdot O\left(\log N \left(\frac{Q^{*4}}{WN^2} + \frac{Q^{*3}}{LN^2}\right)\right) \\
&= O\left(\log N \left(\frac{Q^{*5}}{WN^2} + \frac{Q^{*4}}{LN^2}\right)\right) \\
&= o_N(1)
\end{aligned} \tag{7.15}$$

with our choices of L, W and Q^*. This finishes the proof of the lemma. □

Looking back regarding our choices of $L := (2N)^{2/9}$ and $W := 2N/L = (2N)^{7/9}$, we need L, W and Q^* to satisfy the following two inequalities for the proof to work: $LQ^* = o(W)$ (6.8), and that (7.15) above is $o_N(1)$. Our choices of L and W are optimized to maximize the query complexity Q^* under these conditions.

8 Finding cycles in Bender-Ron-graphs using $O(N^{13/18})$ queries

Given the lower bound established above for cycle-finding in Bender-Ron graphs, one natural question concerns the limitations of this approach: what is the true query complexity of cycle-finding in graphs drawn from this distribution? This section sketches two algorithmic approaches that find cycles with high probability in random graphs $\boldsymbol{G} \sim \mathbb{BR}$ for many values of the length parameter L. In particular, setting $L = \Theta(N^{2/9})$ as in our lower bound construction yields an algorithm for cycle finding in $\mathbb{BR}$ graphs with query complexity roughly $N^{13/18}$.

Algorithm 1. We begin with the following simple observation: With high probability over a random Bender-Ron graph $\boldsymbol{G} \sim \mathbb{BR}$, for each vertex $v \in [3N]$ it is possible to correctly determine the color (and layer, if the color is red) of v in $O(L)$ queries. This is a straightforward consequence of the following two facts. First, if v is a red vertex in layer R_i, then every directed path from v reaches a sink after exactly $L - i$ edges. Second, for almost every graph $\boldsymbol{G} \in \mathbb{BR}$, a sequence of random walks made from any blue vertex in $\boldsymbol{G}$ will differ significantly in the distance they travel before they find a sink. Thus an algorithm can determine the color and layer of v with high probability by making several random walks of length $O(L)$.

We can leverage this observation to find a cycle with high probability in $O(L\sqrt{N})$ queries as follows. The algorithm works by first identifying a blue vertex in $O(L)$ queries by randomly sampling and confirming its color using the procedure described above. Each child of a blue vertex $\boldsymbol{G} \sim \mathbb{BR}$ is blue with probability $1/2$,

so we can grow a blue path from our seed vertex at a cost of roughly $O(L)$ queries to confirm the color of each additional vertex.[11] We construct a blue path of length $C\sqrt{N}$, at which point each successive blue vertex added to the path creates a cycle with probability at least $C/(2\sqrt{N})$. By a birthday paradox argument, the next $C\sqrt{N}$ blue vertices added to the path yield a cycle with high probability for large C. Setting $L = (2N)^{2/9}$ as in our lower bound proof, the query complexity of the resulting algorithm is $O(N^{13/18})$.

Algorithm 2. Algorithm 1 provides a good upper bound on the query complexity of cycle-finding in Bender-Ron graphs when L is relatively small. In this section we sketch a more sophisticated strategy that gives a query-efficient algorithm when L is large. The key observation here is that for almost every graph $\boldsymbol{G} \sim \mathbb{BR}$, given any red vertex v in layer R_i, by making $P := \tilde{O}(W)$ queries it is possible to query almost every vertex in layer R_{i+t}, where $t = \log_d P$. This is accomplished by performing a breadth-first search of depth t starting from vertex v. We think of this as the query algorithm "building a wall" at layer R_{i+t}.

Algorithm 2 has two stages. In the first stage, it builds a series of walls, effectively mapping out the structure of G. In the second stage, it exploits its knowledge about the structure of G to efficiently build a long blue path using a method similar to Algorithm 1.

In more detail, the first stage starts by first identifying M red vertices by sampling vertices at random and using random walks to confirm their color, a process which takes $O(LM)$ queries. In the rest of the first stage the algorithm then performs the wall-building procedure at each of these vertices, a process which takes roughly $\tilde{O}(MW)$ queries.[12] At this point, the query algorithm has built $\Theta(M)$ walls, which with high probability are typically spaced at intervals roughly $O(L/M)$ apart throughout the layers $R_1, \ldots, R_L$. Thus the number of queries that the first stage takes is about

$$\tilde{O}(M(W + L)) = \tilde{O}(M(N/L + L)).$$

In the second stage, Algorithm 2 is the same as Algorithm 1, except that Algorithm 2 can identify vertex colors by reaching a wall instead of a sink vertex. Consider a random walk from a vertex v. If v is in layer R_i, then most random walks from v will collide with the next wall in a particular, fixed number of queries a_i, which will typically be $O(L/M)$. If v is a blue vertex, then most random walks from v will still collide with a wall in $O(L/M)$ queries, but with high probability the length of these walks will vary significantly. As a result, using the same method as Algorithm 1, Algorithm 2 can identify vertex colors in $O(L/M)$ queries and find a cycle of length $O(\sqrt{N})$ in $O(L\sqrt{N}/M)$ queries. Thus the query complexity of Algorithm 2 is

$$\tilde{O}(M(N/L + L) + L\sqrt{N}/M).$$

If $L \gg N^{1/4}$, then taking

$$M = \frac{N^{1/4}L}{\sqrt{N + L^2}} \gg 1,$$

we get that the query complexity of this second approach is roughly $\tilde{O}(N^{1/4}\sqrt{N + L^2})$, which is $o(N)$ for $N^{1/4} \ll L \leq o(N^{3/4})$.

9 Directions for future work: towards upper bounds

Given our $\tilde{\Omega}(N^{5/9})$ lower bound, it is natural to ask the true query complexity of cycle finding in sparse digraphs that are ϵ-far from acyclic. We conjecture that there is an $o(N)$-query algorithm for this problem, and we pose the problem of finding such an algorithm as a tantalizing goal for future work. We conclude with a few comments towards this goal:

1. Let $\ell = \ell(m, \epsilon)$ be the smallest value such that every m-edge digraph G with the smallest feedback arc set of size at least ϵm must have a cycle of length at most ℓ. Fox [Fox18] has proved that $\ell(m, \epsilon) \leq \tilde{O}(\log m)/\epsilon$. It follows that every bounded-outdegree-d N-vertex digraph that is constant-far from acyclic must contain a cycle of length $\tilde{O}(\log N)$. This structural result may be viewed as a highly efficient *nondeterministic* algorithm (with query complexity $\tilde{O}(\log N)$) for the cycle-finding problem that we consider.

2. It is possible that a simple algorithm based on breadth first search may have sublinear query complexity for cycle-finding in far-from-acyclic bounded-degree digraphs. In more detail, we do not know a counterexample to the following conjecture: "Let $0 < \epsilon < 1$ be a (small) constant. Let $\mathcal{A}'$ be an algorithm which works as follows: for $C = C(\epsilon)$ (a large constant) many repetitions, $\mathcal{A}'$ picks a random vertex v in G and performs a breadth first search out from v until $CN/\log N$ vertices have been explored. When run on any N-vertex graph G that is ϵ-far from acyclic, one of the $C(\epsilon)$ BFSes performed by algorithm $\mathcal{A}'$ finds a

[11] With high probability, the number of red vertices we identify is proportional to the length of the path. If a blue node has no blue children, an event which occurs with probability $1/2^d$, we backtrack to the previous node.

[12] If the algorithm finds a sink while attempting to run a breadth-first search of depth t, this wall fails and the procedure continues.

cycle with constant probability." (We note that by considering the case in which G is a union of d many randomly chosen bipartite matchings, it can be shown that $N/\log N$ cannot be replaced by any function of N that is $o(N/\log N)$.)

Acknowledgements

We thank Jacob Fox for telling us about [Fox18], and several anonymous reviewers for helpful comments on an earlier draft. X.C. is supported by NSF IIS-1838154 and NSF CCF-1703925. R.A.S. is supported by NSF IIS-1838154, NSF CCF-1814873, NSF CCF-1563155, and by the Simons Collaboration on Algorithms and Geometry.

References

[ABG+18] Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018. 1

[AKK19] Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *10th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 6:1–6:20, 2019. 1, 1

[BHPR+18] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. In *Proceedings of the 2018 ACM Conference on Innovations in Theoretical Computer Science*, 2018. 1

[BR02] Michael A. Bender and Dana Ron. Testing properties of directed graphs: acyclicity and connectivity. *Random Struct. Algorithms*, 20(2):184–205, 2002. (document), 1, 1.1, 1.2, 3, 3.1, 3.3, 4.1

[BSS08] Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 393–402, 2008. 1, 2

[CEF+05] Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005. 1

[CPS16] Artur Czumaj, Pan Peng, and Christian Sohler. Relating two property testing models for bounded degree directed graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1033–1045, 2016. 1

[CRT05] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005. 1, 1

[CS09] Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009. 1

[ELRS17] Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017. 1, 1

[ERS18] Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th ACM Symposium on the Theory of Computing*, pages 722–734, 2018. 1

[Fei06] Uriel Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006. 1

[Fox18] Jacob Fox. Personal communication, 2018. 1.1, 1, 9

[Gol17] Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, New York, 2017. 1

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. 1, 2

[GR08] Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008. 1

[GRS11] Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Computing*, 25(3):1365–1411, 2011. 1

[HKNO09] Avinatan Hassidim, Jonathan A Kelner, Huy N Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *Proc. 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 22–31. IEEE, 2009. 1

[HS12] Frank Hellweg and Christian Sohler. Property testing in sparse directed graphs: Strong connectivity and subgraph-freeness. In *Algorithms - ESA 2012 - 20th Annual European Symposium*, pages 599–610, 2012. 1

[HS13] Frank Hellweg and Christian Sohler. Property-testing in sparse directed graphs: 3-star-freeness and connectivity. *CoRR*, abs/1312.0497, 2013. 1

[KSS18] Akash Kumar, C. Seshadhri, and Andrew Stolman. Finding forbidden minors in sublinear time: A $n^{1/2+o(1)}$-query one-sided tester for minor closed properties on bounded degree graphs. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 509–520, 2018. 1, 2

[KSS19] Akash Kumar, C Seshadhri, and Andrew Stolman. Random walks and forbidden minors II: a $\text{poly}(d\epsilon^{-1})$-query tester for minor-closed properties of bounded degree graphs. In *Proceedings of the 51st ACM Symposium on the Theory of Computing*, 2019. 1

[MR09] Sharon Marko and Dana Ron. Approximating the distance to properties in bounded-degree and general

sparse graphs. 5(2):22, 2009. 1

[NO08] Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Proc. 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 327–336. IEEE, 2008. 1

[OR11] Yaron Orenstein and Dana Ron. Testing eulerianity and connectivity in directed sparse graphs. *Theoretical Computer Science*, 412:6390–6408, 2011. 1

[ORRR12] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the 23rd annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1123–1131, 2012. 1, 1

[PR07] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007. 1

[RD85] V. Rödl and R.A. Duke. On graphs with small subgraphs of large chromatic number. *Graphs and Combinatorics*, 1(1):91–96, 1985. 1, 2

[Yao77] A. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. Seventeenth Annual Symposium on Foundations of Computer Science (STOC)*, pages 222–227, 1977. 3

[YYI09] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 225–234. ACM, 2009. 1