Markov Chains with Rewinding

Amir Azarmehr
Northeastern University, Boston, MA, USA

Soheil Behnezhad
Northeastern University, Boston, MA, USA

Alma Ghafari
Northeastern University, Boston, MA, USA

Madhu Sudan
Harvard University, Cambridge, MA, USA

Abstract

Motivated by techniques developed in recent progress on lower bounds for sublinear time algorithms (Behnezhad, Roghani and Rubinstein, STOC 2023, FOCS 2023, and STOC 2024) we introduce and study a new class of *randomized algorithmic* processes that we call "Markov Chains with Rewinding". In this setting an agent/*algorithm* interacts with a (partially observable) Markovian *random* evolution by periodically/strategically rewinding the Markov chain to previous states. Depending on the application this may lead the evolution to desired states faster, or allow the agent to efficiently learn or test properties of the underlying Markov chain that may be infeasible or inefficient with passive observation.

We study the task of identifying the initial state in a given partially observable Markov chain. Analysis of this question in specific Markov chains is the central ingredient in the above cited works and we aim to systematize the analysis in our work. Our first result is that any pair of states distinguishable with any rewinding strategy can also be distinguished with a *non-adaptive* rewinding strategy (i.e., one whose rewinding choices are predetermined before observing any outcomes of the chain). Therefore, while rewinding strategies can be shown to be strictly more powerful than passive strategies (i.e., those that do not rewind back to previous states), adaptivity does *not* give additional power to a rewinding strategy in the absence of efficiency considerations.

The difference becomes apparent however when we introduce a natural efficiency measure, namely the *query complexity* (i.e., the number of observations they need to identify distinguishable states). Our second main contribution is to quantify this efficiency gap. We present a non-adaptive rewinding strategy whose query complexity is within a polynomial of that of the optimal (adaptive) strategy, and show that such a polynomial loss is necessary in general.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Partially observable Markov chains, rewinding algorithm, sublinear algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2026.43

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2602.16028

Funding Amir Azarmehr, Soheil Behnezhad, and Alma Ghafari were supported in part by NSF CAREER Award CCF-2442812 and a Google Research Award. Madhu Sudan was supported in part by a Simons Investigator Award, NSF Award CCF 2152413 and AFOSR award FA9550-25-1-0112.

1 Introduction

In this work, we formally define and study a new class of processes involving interaction between randomness and algorithms. In these processes, that we call *Markov chains with rewinding*, a partially observable Markov chain interacts with an algorithmic agent that has

the ability, at any point in time, to rewind the Markov chain to a previous point of time. We study the problem of *identifying the initial state* in this model, where the goal is for the algorithm to use the observations and the power of rewinding to identify a given (hidden) initial state.

Partially observable Markov chains form a well-established element of the toolkit in computing and arise in both the design and analysis of algorithms. Rewinding of Markov chains captures a natural ability of algorithms that work with Markov chains, and as a result, they have occurred implicitly in a variety of settings (an overview is given in Section 1.2).

This is highlighted centrally in a class of applications of Markov chains with rewinding to *sublinear time algorithms*. For the maximum matching problem, [8, 7, 9] prove strong lower bounds on the query complexity, and hence running time, of sublinear time algorithms by using Markov chains with rewinding to model sublinear algorithms, and then analyzing these chains. Recent works on non-adaptive sublinear lower bounds follow similar paradigms, while not explicitly defining a Markov chain [5, 25]. Additionally, lower bounds for other graph problems in the sublinear-time model, such as edge orientation [22] and acyclicity [11], can also be viewed as instances of this general model.

Despite the prevalence of this phenomenon, rewinding does not appear to have been studied formally in the literature and in particular tools to analyze the power (and limitations) of Markov chains with rewinding have been ad-hoc. Our work is motivated principally by these and aims to build a more systematic toolkit for such analysis.

Markov chains with rewinding. We start with an example to illustrate the problem. Consider the partially observable Markov chain depicted below, where the only observable variable is the color of the current state (i.e., whether it is in $\{a, a', b, b'\}$ or $\{s\}$). The initial state is promised to be either a or a', and our goal is to identify this initial state.

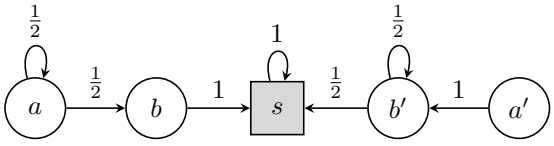

Figure 1.1 Rewinding is stronger than passive observation.

First, note that "passive observations" cannot distinguish a from a': Whether we start from a or a', it takes exactly Geometric$(1/2) + 1$ steps until we reach s and remain there forever. But suppose we are allowed to *rewind* back to any previous state at any time. In this case, we can run the Markov chain for one step to reach state X, and then run D independent steps from X for a sufficiently large D. The idea is that if $X = b'$ (which implies the initial state is a'), then it will most likely have a mix of s and not s next steps, whereas if $X \in \{a, b\}$ (implying that the initial state is a) all next steps will be the same. The following figure illustrates this.

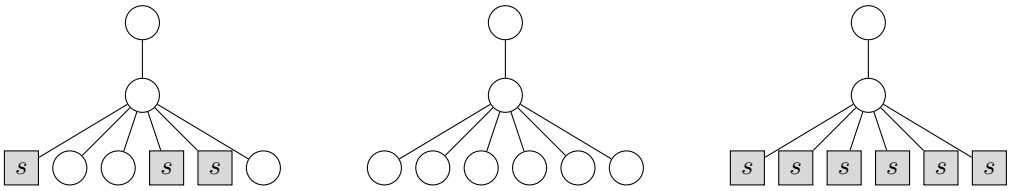

Figure 1.2 Examples of states generated by repeatedly rewinding to the second step.

We now present a more general description of the model, deferring its formalization to Section 3. The input consists of a partially observable Markov chain M, two candidate initial states a and a', and a hidden initial state $X_0 \in \{a, a'\}$. The algorithm interacts with the Markov chain through the hidden initial state X_0. It can either draw states according to the transition probabilities, or rewind the state to an earlier point in time. The drawn states $\{X_t\}_t$ are hidden from the algorithm, but observations $\{Z_t\}_t$ are available, where $Z_t = O(X_t)$. Here O is a fixed observation function mapping each state to its observation, so that $O(X_t)$ is the observation revealed at time t. Eventually, the algorithm must decide whether the initial state X_0 is a or a', based on the observations. We study the runtime and the number of queries (i.e. drawn states) required to do so. Other than the motivation from sublinear-algorithm lower bounds, rewinding formalizes a natural approach to interacting with Markov chains; variations and special cases have been considered repeatedly in previous work (see Section 1.2).

1.1 Our Contributions

Our main focus in this work is on the following fundamental problem:

> *Given a partially observable Markov chain, can a nearly-optimal rewinding strategy be computed efficiently?*

To better understand this question, we first need to formalize the model. We present the formalization in Section 3; We define *rewinding strategies* and *state identification algorithms*. Rewinding strategy refers to the choice of rewinding times throughout the process. A state identification algorithm computes the rewinding strategy through which it interacts with the partially observable Markov chain, and ultimately identifies the initial state based on the observations. We study and compare the power of *adaptive* and *non-adaptive* rewinding strategies for this task, where a non-adaptive strategy determines how to rewind independently of the observations. These lead us to our key complexity measures $\mathsf{QC}_M(a, b)$ (resp. $\mathsf{NAQC}_M(a, b)$) for the adaptive (resp. non-adaptive) *query complexity* of distinguishing state a from b in chain M, i.e. the minimum number of observations required by a rewinding strategy to distinguish the two states. Query complexity provides a benchmark for measuring the efficiency of an algorithm, and a means of proving lower bounds for its runtime.

Crucially, the algorithm is given the full description of the chain $M = (\Omega, P, O)$, here Ω is the (finite) state space of the Markov chain, $\Omega \times \Omega \to [0, 1]$ is its transition probability matrix, and O is the observation function introduced above. What is hidden is only the realized trajectory $\{X_t\}_t$ and the initial state.

We show that there is indeed a polynomial-time algorithm for the question stated at the beginning of this section. Formally, we prove the following theorem.

▶ **Theorem 1.1.** *Given a partially observable Markov chain $M = (\Omega, P, O)$, there exists a non-adaptive algorithm and a constant $c = c(|\Omega|)$ that can distinguish between any two states $a, b \in \Omega$ in time*

$$c \cdot \mathsf{QC}_M(a,b)^{O(|\Omega|)}.$$

Note that we are using the query complexity as our measure of efficiency. That is, we devise a state identification algorithm that chooses the rewinding states in time $\text{poly}(\mathsf{QC}_M(a,b))$ and generates at most $\text{poly}(\mathsf{QC}_M(a,b))$ states. In the statement above, we let M be a Markov chain on state space Ω with a special *sink* state. The only observable signal available to the rewinding algorithm is whether the chain is in the sink state or not (similar to the example in Section 1). We call such Markov chains *canonical* and show that identifying the initial state in canonical Markov chains is essentially as hard as general partially observable Markov chains (see Section 6).

Observe that if non-adaptive algorithms cannot distinguish two states a and b of a Markov chain M, then Theorem 1.1 implies that no adaptive algorithm can do so either. Therefore, non-adaptive algorithms are as powerful as adaptive ones in terms of which pairs of states they can distinguish.

Furthermore, formalized as Theorem 1.2, we show that the power of $\mathsf{QC}_M(a,b)$ in Theorem 1.1 is tight, up to a constant, for the runtime of non-adaptive algorithms. More specifically, we present a partially observable Markov chain M and states a and b, where a polynomial gap exists between the adaptive and non-adaptive query complexity of distinguishing a and b.

▶ **Theorem 1.2.** *There is a choice of M and pair $a, b \in \Omega$ such that for some $c = c(|\Omega|)$,*

$$\mathsf{NAQC}_M(a,b) \geq c \cdot \mathsf{QC}_M(a,b)^{(1-o(1))|\Omega|}.$$

This holds for infinitely many choices of $|\Omega|$ and query complexities.

We remark that the adaptive rewinding strategy presented for this gap can be easily turned into an algorithm with the same run time. Therefore, the polynomial gap is also present between the optimal adaptive and non-adaptive run times of state identification algorithms.

We leave open the question of whether the number of states in the Markov chain can be removed from the run-time exponent. Any solution would require new techniques and necessarily rely on adaptivity, as remarked above. Formally, we ask:

▶ **Open Problem 1.** *Given a partially observable Markov chain $M = (\Omega, P, O)$, candidate initial states $a, b \in \Omega$, and hidden initial state $X_0 \in \{a, b\}$, is there an adaptive state identification algorithm that successfully identifies the initial state in time $\mathsf{QC}_M(a,b)^C$ where constant C is independent of the number of states?*

We believe it is also interesting to compare non-adaptive algorithms against the best non-adaptive benchmark for future work. In particular, we ask:

▶ **Open Problem 2.** *Given a partially observable Markov chain $M = (\Omega, P, O)$, candidate initial states $a, b \in \Omega$, and hidden initial state $X_0 \in \{a, b\}$, is there a non-adaptive state identification algorithm that successfully identifies the initial state in time $\mathsf{NAQC}_M(a,b)^C$ where constant C is independent of the number of states?*

1.2 Related Work

We overview settings where partially observable Markov chains with rewinding have occurred naturally. A highly prevalent instance is the folklore method of converting expected *run times of algorithms* into high probability bounds by restarting the algorithm when it does not halt promptly. This is a special case of rewinding where the underlying state of the algorithm is captured by a Markov chain, and the observation is limited to knowing whether the algorithm has halted or not.

Restarting/rewinding can also help boost the success of algorithms as exemplified in the famed randomized algorithm for 3SAT due to Schöning [24]. In the case of random walks on multi-dimensional lattices, [21] showed that the expected hitting time of the origin is finite when the algorithm is allowed to restart the walk back from the starting point. More general forms of rewinding lead to elegant new approaches to *optimization* and *random sampling* as in the "Go-with-the-winners" algorithm of Aldous and Vazirani [1]. Selecting which state to continue the process from is also studied by [18], where the algorithm decides which Markov chain to advance, in order to minimize the expected time that one of them reaches the target state. In *cryptography*, rewinding arguments play a celebrated role in proving the security of protocols starting with the seminal work of Goldwasser, Micali, and Rackoff [20]. Rewinding is also a central element in defining strong notions of security as in the notion of "resettable zero knowledge" due to Canetti, Goldreich, Goldwasser, and Micali [13].

A related line of work studies other forms of controlled Markov chains. For instance, Azar, Broder, Karlin, Linial, and Phillips [4] introduced biased random walks, in which a controller may, with small probability, override the random transition to steer the walk, and gave tight bounds and polynomial-time algorithms for the optimal control strategy maximizing a long-term benefit.

1.3 Connection to Sublinear-Time Graph Algorithms

To further motivate our model, we overview its connection to sublinear time graph algorithms (or more precisely, lower bounds). Here, we provide a high-level description of this connection in general. Later in Appendix A, we present an explicit connection to the lower bound for testing acyclicity. That is, we show how the lower bound is captured by Markov chains with rewinding.

Let us first briefly overview the model of sublinear-time algorithms for graphs. We are given a graph $G = (V, E)$ to which we have adjacency list query access.[1] Each query can specify a vertex v and an integer $i \geq 1$. The answer to such a query is the i-th neighbor of vertex v stored in an arbitrarily ordered list, or $\perp$ if $\deg(v) < i$. Now, the main question of interest, is the number of such queries needed to estimate a property of the graph. For example, the works of [8, 7, 9] focus on proving lower bounds on the query complexity of estimating the size of maximum matching in the graph (see also [6, 26, 23, 12] for some algorithms). Many other problems can, and have been, studied in this setting, including graph coloring [2], correlation clustering [3], metric properties such as minimum spanning trees [17, 14], Steiner trees [15], TSP [16, 10], and many others.

Remarkably, for many graph properties, we have had *unconditional* query lower bounds that can be matched algorithmically with sublinear time algorithms. However, these lower bound arguments are often ad hoc, and there is a lack of systematic tools to prove them. Our hope is that our model and techniques can serve as a first step towards developing such generic lower bound tools. Let us expand on this connection a bit further.

[1] Another common access model is the adjacency matrix model.

Typically, sublinear time lower bounds are proved by specifying two distributions $\mathcal{D}_{YES}$ and $\mathcal{D}_{NO}$ of graphs, where graphs drawn from $\mathcal{D}_{YES}$ have the desired property and those drawn from $\mathcal{D}_{NO}$ do not. The lower bound then follows by showing that distinguishing these distributions requires many queries to the graph. In most cases, these distributions are constructed by having a few groups $A_1, A_2, \ldots, A_k$ that each vertex may belong to and is kept hidden from the algorithm. The distribution of edges between these groups differs in $\mathcal{D}_{YES}$ and $\mathcal{D}_{NO}$, and thus the task of distinguishing these distributions reduces to identifying these groups. The only useful signal that the algorithm receives, and helps in identifying the groups, is the vertex degrees. This is precisely how the hard instances of [8, 7, 9] for maximum matchings are constructed. Lower bounds for other graph problems in the sublinear-time model, such as edge orientation [22] and acyclicity [11], can also be viewed as instances of this general framework.

Now note that each group A_i can be viewed as a different state of the Markov chain. We let p_{ij} (i.e., the transition probability from A_i to A_j) to be the fraction of edges of group A_i that go to A_j, representing the fact that a random adjacency list neighbor of an A_i node belongs to A_j with probability p_{ij}. We construct these states separately for groups in $\mathcal{D}_{YES}$ and $\mathcal{D}_{NO}$. Then, assuming that the algorithm cannot find cycles (which holds in many lower bounds such as [8, 7, 19, 11]) the task of distinguishing the two distributions reduces to distinguishing if the initial state is among the YES states or the NO states of the Markov chain. Note, in particular, that because the sublinear time algorithm is not constrained to follow a single path in the graph and can adaptively make adjacency list queries to any previously visited vertex, this corresponds to a rewinding strategy on the Markov chain. Therefore, the study of the power and limitations of such rewinding strategies for general Markov chains closely relates to the query complexity of sublinear-time graph algorithms, and can serve as a rich toolkit for proving lower bounds against them.

Paper Arrangement

More examples of Markov chains with rewinding are presented in Section 2. An overview of our techniques appears in Section 2.3. The model is formally defined in Section 3. We examine state partitions, a key tool for our algorithm, in Section 3.1. Proofs of Theorems 1.1 and 1.2 are sketched in Sections 4 and 5 respectively. Finally, in Section 6, we prove the generality of canonical Markov chains. For the full proofs refer to the following version [5].

2 Motivating Examples & Technical Overview

We start this section by presenting examples of rewindable Markov chains along with optimal strategies to distinguish between two candidate initial states. These examples further clarify the model and demonstrate how rewinding strategies can be *non-trivially* effective, and thus hard to prove lower bounds against. After these examples, we outline the techniques used in our algorithms and analyses.

2.1 Example 1: The non-trivial power of (non-adaptive) rewinding

Our first example involves a simple Markov chain on 5 states, inspired by the hard distribution of [8] for graph matchings. The chain is parameterized by an integer d, and its transition probabilities are multiples of $1/d$ as illustrated by the arrows in the figure. There is a special sink state s, and the algorithm can only observe whether a state is sink or non-sink. The goal is to distinguish whether the initial state is a or a'. We show first how the natural strategy to distinguish two candidate initial states a and a' leads to query complexity $\widetilde{O}(d^2)$. A quick look at the chain and the rewinding strategy seems to suggest that such a d^2 complexity is

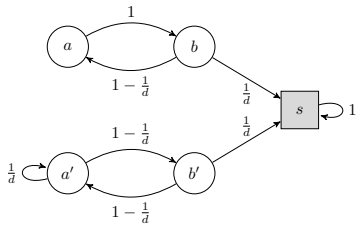

Figure 2.1 A five-state example.

essential; but we show later that a more sophisticated strategy actually distinguishes a from a' in $O(d)$ queries.

A simple algorithm with $\widetilde{O}(d^2)$ query complexity. We say a node in the tree is an A-node if its state is a or a' and a B-node if its state is b or b'. First, we claim that given a node v in the tree, we can determine with probability $1-\varepsilon$ if it is an A or B node with $O(d\log(1/\varepsilon))$ queries – we call this an A/B test. To do this, note that if v is an A-node, any child of v is non-sink (specifically, it is a B-node). Moreover, if v is a B-node, each child has a probability of $1/d$ of being a sink node. Therefore, opening $O(d\log(1/\varepsilon))$ children for a node v is sufficient for A/B testing v.

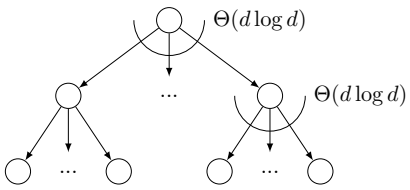

Figure 2.2 An $O(d^2\log^2 d)$-query strategy.

Next, having the A/B test available to us, we aim to assert whether a given A-node v has state a or a'. To do so, we first open $d\log(1/\varepsilon)$ children for v. If v has state a, then all of its children must be B-nodes. On the other hand, if v has state a', each child is an A-node with probability $1/d$, and so at least one must be an A-node with probability $1-(1-1/d)^{d\log(1/\varepsilon)} \geq 1-\varepsilon$. It then suffices to run the A/B test on these children to check if any of them is an A node. This takes total time $O(d^2\log^2(1/\varepsilon))$. Choosing $\varepsilon = 1/d^3$ so that all A/B tests are correct w.h.p., we get an overall query complexity of $O(d^2\log^2 d)$. The final queried tree is non-adaptive and is illustrated on the right.

Improving query complexity to $O(d)$. We now describe how we can improve over the algorithm above, and distinguish the a and a' states with just $O(d)$ queries. Here is the key insight. Suppose that we open a path of length $2d$ from the root and let us condition on not seeing the s node at all (which happens with constant probability). Then, if we start from the a state, we must alternatively visit a and b, finally arriving back at a since the length of the path is even. However, if we start from a', there is a constant probability that we go from a' to a' exactly once throughout the process since $P(a',a') = 1/d$ and the length of the walk is $2d$. If this event happens, we end up in state b'. Therefore, it suffices to run the A/B test on the last vertex of the path – if it happens to be B, we will be sure that we started from a'. Note that the success probability is a small constant, but we can boost it by

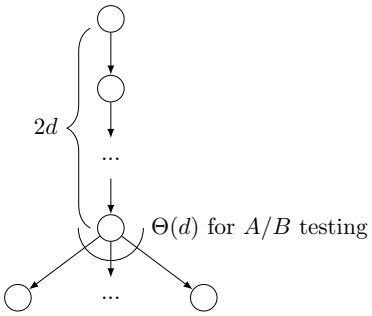

Figure 2.3 An $O(d)$-query strategy.

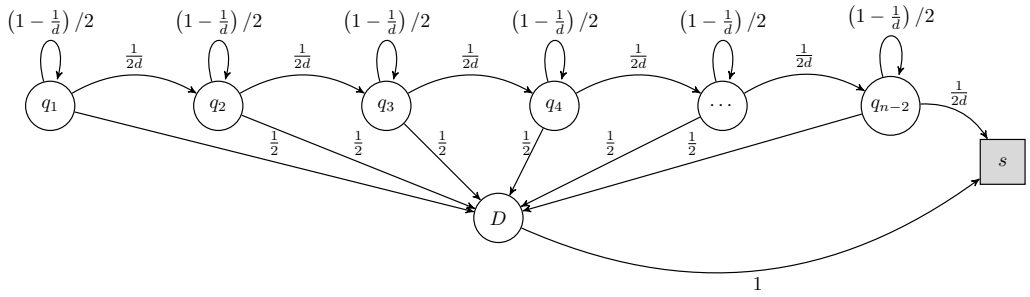

Figure 2.4 A canonical Markov chain for which there is a (large) polynomial gap between adaptive and non-adaptive strategies of distinguishing states q_1 and q_2.

repeating this process multiple times. The figure on the right shows the final query tree (for a single repetition of this process), which again is non-adaptive.

2.2 Example 2: The power of adaptivity

Our second example shows the power of adaptive rewinding strategies compared to non-adaptive ones. This example, illustrated in Figure 2.4, is later used to prove a polynomial gap between the non-adaptive and adaptive query complexities in the worst case (Theorem 1.2). We formalize this in Section 5, but here provide an intuition of how adaptivity helps.

Given a parameter $d \geq 1$, the Markov chain is formed by combining a "simple Markov chain" and a "dummy state". The simple Markov chain consists of a path of states, where each state transitions to the next with probability $\frac{1}{d}$, and stays in the same state otherwise. States can be easily distinguished in the simple Markov chain using $O(n^2 d)$ queries, where $n = |\Omega|$ is the number of states. To do so, it suffices to take random walks from the initial state and examine the time it takes to move to the sink (see Section 5 for the proof).

However, when combined with the dummy state, every state has a $\frac{1}{2}$ chance of jumping to the dummy state (otherwise, it proceeds as before). Then, the dummy state directly moves to the sink with probability 1. As all the states have the same probability of transitioning to the dummy state, moving to the sink through the dummy state reveals no information about the initial state.

Adaptive strategies can easily filter out the dummy state. That is, they can simulate a random walk on the simple Markov chain by checking if the walk has transitioned to the dummy state along the way, and rewinding back to the previous step if it has. As a result,

adaptive strategies retain the $O(n^2 d)$ query complexity even with the dummy state added.

The non-adaptive strategies cannot do the same, i.e. they cannot check whether a random walk has transitioned to the dummy state on the fly (they can only infer it after all the queries are made). Intuitively, to make up for this, they have to take many more random walks from the initial state, so as to guarantee that plenty of them reach the sink without moving through the dummy state. Therefore, the non-adaptive query complexity suffers when the dummy state is added and becomes (approximately) $\Omega(d^n)$. See Section 5 for the proof.

2.3 Technical Overview

Recall that our main goal is to bring a systematic understanding of distinguishing two candidate initial states in a known Markov chain. In this section, we provide a brief overview of our methods. There are two main aspects associated with our task: (1) computing rewinding strategies to distinguish between two initial states, and (2) proving lower bounds for the query complexity of this task. Together the two aspects allow us to establish the efficiency of our algorithm, i.e. that its run time is competitive with the optimal query complexity. Our approach to both aspects heavily utilizes partitions of the state set Ω.

Let $\mathcal{P}$ be a partition of the states, and for a state $a \in \Omega$ let $\mathcal{P}(a)$ denote the "class" of a in $\mathcal{P}$, i.e., the set within the partition $\mathcal{P}$ that contains a. As a subroutine, we study the problem of identifying the class of any given (hidden) state in $\mathcal{P}$, i.e., the task of computing $\mathcal{P}(X_0)$ for a given chain M and partition $\mathcal{P}$ (note that this is a generalization of the A/B test in Section 2.1). By choosing $\mathcal{P}$ carefully, this allows us to distinguish between two states $a, b \in \Omega$. Specifically, it suffices to identify the class of the initial state for some partition $\mathcal{P}$ that separates a and b, i.e., where $\mathcal{P}(a) \neq \mathcal{P}(b)$.

Intuitively, identifying the class within a given partition becomes more difficult as the partition becomes more refined. As an extreme case, the test for the trivial partition $\mathcal{P}_0 = \{\Omega \setminus \{s\}, \{s\}\}$ can be performed without any further queries in a canonical Markov chain since the partial observation at any state determines its class in $\mathcal{P}_0$. Our key tool in the design of algorithms is to assume that for some partition $\mathcal{P}_1$ we know how to identify the states, and then to use this algorithm to build a more refined partition $\mathcal{P}_2$ where we can identify states. Starting this with the trivial partition, we can use this method to work our way up to more and more refined partitions, till we reach a partition that separates a and b at which stage our problem would be solved. The exact details of this refinement step (and how we estimate their query complexity) are described below.

Recall that the goal of Theorem 1.1 is to develop a non-adaptive algorithm that distinguishes between two states a and b in time $O_n(1) \cdot \mathsf{QC}_M(a,b)^{O(n)}$, where n is the number of states and $O_n(1)$ suppresses terms dependent only on n. To do so, we define a weighted directed graph on the partitions: Namely, every partition $\mathcal{P}$ of Ω is a vertex of this graph and there is a weighted edge between every pair of partitions $\mathcal{P}_1 \to \mathcal{P}_2$ where $\mathcal{P}_2$ refines $\mathcal{P}_1$. The weights $w(\mathcal{P}_1, \mathcal{P}_2)$ are formally defined shortly. Intuitively, the main idea is to get a quantity that roughly allows us to determine the class of the initial state in $\mathcal{P}_2$ using approximately $w(\mathcal{P}_1, \mathcal{P}_2)$ calls to the class-identifier for $\mathcal{P}_1$. Note that if there is a path $\mathcal{P}_1 \to \mathcal{P}_2 \to \mathcal{P}_3$ in this graph, then this amounts to saying that classes of $\mathcal{P}_3$ can be identified using roughly $w(\mathcal{P}_1, \mathcal{P}_2) \cdot w(\mathcal{P}_2, \mathcal{P}_3)$ calls to the class identifier for $\mathcal{P}_1$. Thus, given this graph, we can easily devise a non-adaptive strategy for distinguishing between a and b.

For every partition $\mathcal{P}$ we compute the minimum distance from the trivial partition $\mathcal{P}_0$ (using this multiplicative measure of length of a path), and then use the partition $\mathcal{P}$ that separates a from b and has the smallest distance among all such $\mathcal{P}$. In particular, if we

let $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k = \mathcal{P}$ be the shortest path to $\mathcal{P}$, the test for each $\mathcal{P}_i$ can be recursively performed by calling the tests $\mathcal{P}_{i-1}$. This results in a test for $\mathcal{P}_k$ (which recall differentiates between a and b) that uses $c(\mathcal{P}_k) = \prod_{1 \le i \le k} w(\mathcal{P}_{i-1}, \mathcal{P}_i)$ tests. For Theorem 1.1, we also need to prove $c(\mathcal{P}_k) \le \mathsf{QC}_M(a,b)^{O(n)}$ which we discuss shortly, but first we describe how we compute $w(\mathcal{P}_1, \mathcal{P}_2)$.

To understand how a test for $\mathcal{P}_1$ can be used to identify the class in $\mathcal{P}_2$ of a given state, consider, as a warm-up, the original problem of distinguishing between a and b. Equipped with the test for $\mathcal{P}_1$, one can differentiate between a and b as follows: Given a state $x \in \{a, b\}$, we take many samples from the next state (rewinding back to x after each sample) and examine the distribution of their classes in $\mathcal{P}_1$. If the distribution of these classes for a and b are δ-apart in total variation distance, then x can be identified successfully with probability $1 - \varepsilon$ using $O(\log(1/\varepsilon)/\delta^2)$ samples. A similar approach can be taken to distinguish classes of $\mathcal{P}_2$ (instead of two states a and b). This is again captured by the total variation distance of certain distributions which gives us the weight $w(\mathcal{P}_1, \mathcal{P}_2)$ in the graph. Specifically, we can take

$$w(\mathcal{P}_1, \mathcal{P}_2) = \max_{\substack{c,d: \\ \mathcal{P}_1(c)=\mathcal{P}_1(d) \\ \mathcal{P}_2(c)\neq\mathcal{P}_2(d)}} \frac{1}{d_{\mathrm{TV}}^{\mathcal{P}}(c,d)^2},$$

where for $x \in \{c, d\}$, and $d_{\mathrm{TV}}^{\mathcal{P}}(c,d)$ denotes the total variation distance between next states from c and d when projected on $\mathcal{P}$. That is, letting X^c and X^d be the next state of the Markov chain conditioned on the last state being c and d respectively, $d_{\mathrm{TV}}^{\mathcal{P}}(c,d) := d_{\mathrm{TV}}(\mathcal{P}(X^c), \mathcal{P}(X^d))$. Thus, this expression captures the hardest to separate pair of states c and d from different classes of $\mathcal{P}_2$ (and not already separated in $\mathcal{P}_1$) after taking one step on the Markov chain starting at these states, using only a class identifier for $\mathcal{P}_1$.

Turning to our lower bound, recall that we wish to express the shortest path length in terms of the query complexity, i.e. to prove that $\prod_{1 \le i \le k} w(\mathcal{P}_{i-1}, \mathcal{P}_i)$ is at most

$$O_n(1) \cdot (\mathsf{QC}_M(a,b))^{O(|\Omega|)},$$

where $\mathcal{P}_0 \to \mathcal{P}_1 \cdots \to \mathcal{P}_k$ is the shortest path found by our algorithm. The argument has two main components, given a partition $\mathcal{P}$: (1) if two states c and d with $\mathcal{P}(c) = \mathcal{P}(d)$ are far in terms of $d_{\mathrm{TV}}^{\mathcal{P}}$, then $\mathcal{P}$ has a small outgoing edge separating c and d, and (2) if every two states within the same class of $\mathcal{P}$ are close in terms of $d_{\mathrm{TV}}^{\mathcal{P}}$, then it is difficult to distinguish any two states in the same class of $\mathcal{P}$. Note, in particular, that if such a $\mathcal{P}$ does not separate a and b, then it would serve as a "certificate" for the difficulty of distinguishing between them.

We begin by explaining how these components can be applied to obtain upper bounds on the shortest path (equivalently, lower bounds on the query complexity). Consider any partition $\mathcal{P}$, where a and b are not separated. Given that a and b are distinguishable with $\mathsf{QC}_M(a,b)$ queries, we can employ (2) to show there are two states c and d with $\mathcal{P}(c) = \mathcal{P}(d)$ such that $d_{\mathrm{TV}}^{\mathcal{P}}(c,d) = \Omega(1/\mathsf{QC}_M(a,b))$. Then, using (1), we can infer that $\mathcal{P}$ has an outgoing edge of weight at most $O_n(1)/d_{\mathrm{TV}}^{\mathcal{P}}(c,d)^2 = O_n(1) \cdot \mathsf{QC}_M(a,b)^2$. As a result, we can start at $\mathcal{P}_0$ and iteratively take the smallest outgoing edge, which has weight at most $O_n(1) \cdot \mathsf{QC}_M(a,b)^2$, until we reach a partition that separates a and b. The path can take a length of at most $n-1$ since each partition is refined by the next. This gives us a path of length at most $O_n(1) \cdot \mathsf{QC}_M(a,b)^{2n}$ from $\mathcal{P}_0$ to a partition that separates a and b.

Now we discuss each component separately. First, consider a partition $\mathcal{P}$ with two states c and d in the same class of $\mathcal{P}$ that have $d_{\mathrm{TV}}^{\mathcal{P}}(c,d) = D$. Let G be an auxiliary graph where

the vertices are the Markov chain states, and two states are connected if they are closer than D/n in $d_{\mathrm{TV}}^{\mathcal{P}}$. We construct a refinement $\mathcal{P}'$ of $\mathcal{P}$ by letting two states be in the same class of $\mathcal{P}'$ if they are in the same class of $\mathcal{P}$ and the same connected component of G. Observe that $\mathcal{P}'$ separates c and d since $d_{\mathrm{TV}}^{\mathcal{P}}$ forms a metric, and c and d are more are D apart, whereas any two directly connected states are at most D/n apart. Furthermore, the only states that are newly separated by $\mathcal{P}$ are at least D/n apart. Therefore, $\mathcal{P}$ has an outgoing edge of weight at most $(n/D)^2$ to $\mathcal{P}'$.

For the second component, take a partition $\mathcal{P}$ where any two states in the same class of $\mathcal{P}$ are at most θ apart in $d_{\mathrm{TV}}^{\mathcal{P}}$. We show that for any two states a and b with $\mathcal{P}(a) = \mathcal{P}(b)$, it holds $\mathsf{QC}_M(a, b) = \Omega(1/\theta)$. Given two possible initial states a and b to distinguish, we consider two instances of running the rewinding strategy: one with a, and one with b. Let T_a and T_b be the query trees that the rewinding strategy creates in each case. We couple T_a and T_b such that they look the same to the strategy with a large probability. Here, by looking the same, we mean that the states generated in T_a and T_b yield the same observations, i.e. they are both sink or non-sink. In addition, the coupling is such that T_a and T_b are isomorphic w.r.t. $\mathcal{P}$, meaning the corresponding nodes in T_a and T_b have states that are in the same class of $\mathcal{P}$, with a large probability.

We construct our couplings in an inductive manner. In every step, a state $u_a \in T_a$ and the corresponding state $u_b \in T_b$ are chosen by the rewinding strategy to derive the next step. At this point, we need a method to couple the next step drawn from u_a and u_b such that they are both sink or non-sink with a large probability. Such a coupling is not possible for arbitrary states u_a and u_b (e.g. $\mathcal{P}_0(u_a) = \mathcal{P}_0(u_b)$ does not imply the same is true for their next step with a large probability). To do so, we employ the additional property that $\mathcal{P}(a) = \mathcal{P}(b)$, and show that the drawn states can be coupled such that they are in the same class of $\mathcal{P}$ with probability $1 - \theta$. Intuitively, if a rewinding strategy distinguishes between a and b, it should make at least $\Omega(1/\theta)$, so that the above scheme fails and the drawn states are in different classes of $\mathcal{P}$.

We remark that something as strong as the shortest path algorithm is not required for Theorem 1.1. While it is intuitive to use the shortest path as it corresponds to "the cheapest way" for testing $\mathcal{P}$ in our graph, something as simple as Prim's algorithm also works. That is, to prove Theorem 1.1, we essentially show that there exists a partition $\mathcal{P}$ separating a and b which is reachable from $\mathcal{P}_0$ through edges of weight at most $O_n(1) \cdot \mathsf{QC}_M(a, b)^2$. Therefore, Prim's algorithm can find such a $\mathcal{P}$, by starting from $\mathcal{P}_0$ and in each step, adding the minimum outgoing edge to the tree. Also, note that while the shortest path algorithm may use fewer queries in many cases, in the worst case, it uses (asymptotically) the same number of queries as Prim's algorithm (Theorem 1.2).

3 The Formal Model

We use $\mathbb{Z}^{\geq 0}$ to denote the set of non-negative integers. We use $d_{\mathrm{TV}}(\cdot, \cdot)$ to denote the total variation distance between two distributions.

We start by reviewing the standard notions of (partially observable) Markov chains. A sequence of random variable $X_0, X_1, X_2, \ldots$ with $X_t \in \Omega$ for all t is a *Markov chain* if

there exists a transition matrix $P : \Omega \times \Omega \to [0, 1]$ such that for every t and all $x, y \in \Omega$, $\Pr(X_{t+1} = y \mid X_t = x) = P(x, y)$; we write $P(x, \cdot)$ for the distribution from x to the next state. [2] A sequence $Z_0, Z_1, \ldots$ with $Z_t \in \Sigma$ is a *partially observable (p.o.) Markov chain* if

[2] We note that such Markov chains are called *time-invariant* since transition probabilities, conditioned on the current state, are independent of time.

there exists a Markov chain $X_0, X_1, \ldots$ and a function $O : \Omega \to \Sigma$ such that $Z_t = O(X_t)$ for all t. (While a general theory would allow probabilistic functions O, we restrict our attention to deterministic functions in this work.) We refer to Ω as the state space and Σ as the observable space. Note that a partially observable chain M is given by a triple (Ω, P, O).

Next, we turn to the central objects of study in this paper, namely Markov chains with rewinding. Before doing so, we briefly touch upon two views of a rewinding strategy. The formal view is simply as a function that maps a history of observations to a non-negative number (which we view as the number of steps we rewind the current time by). An alternate view is that the rewinding strategy builds a tree where, at each time step, the algorithm picks a node of the current tree, and the next time step produces a new child for this node whose state is independent of that of all other nodes, conditioned on the parent's state. The following definition uses the formal view, but we often switch to the tree view in our analyses.

▶ **Definition 3.1** (Markov Chains with Rewinding). *For a partially observable Markov chain $M = (\Omega, P, O)$ with observable space Σ, a rewinding strategy is a function $A : \Sigma^* \to \mathbb{Z}^{\geq 0}$. A Markov chain $M = (\Omega, P, O)$ in initial state X_0 with rewinding strategy A generates a sequence of random variables $Z_0, Z_1, Z_2, \ldots$, with companion variables $X_1, X_2, \ldots$ as follows: For every $t \geq 0$, we let $Z_t = O(X_t)$ and $X_{t+1} \sim P(X_{t'}, \cdot)$ where $t' = \max\{0, t - A(Z_0, \ldots, Z_t)\}$. We say that a sequence of random variables $Z_0, Z_1, \ldots,$ is a Markov chain with rewinding if there exists a chain M, an initial state X_0, and a rewinding strategy A that generates this sequence.*

Markov chains with rewinding strictly generalize the class of Markov chains, which correspond to *passive observation*, i.e., the rewinding strategy $A(\cdot)$ is the constant function 0, or equivalently, the rewinding tree is a path (and hence "chain"). Another class of rewinding strategies, widely used in algorithm design, picks some threshold $\tau \in \mathbb{Z}^{\geq 0}$ and sets $A(Z_0, \ldots, Z_t) = t$ if $t = 0 \pmod{\tau}$ and $A(Z_0, \ldots, Z_t) = 0$ otherwise. Equivalently, the rewinding tree consists of paths of length τ intersecting at the root. Such strategies may be termed *resetting strategies*. A simple adaptation of resetting is inspired by the "go-with-the-winners" algorithm [1] is the following: $A(Z_0, \ldots, Z_t) = 0$ if $t \neq 0 \pmod{\tau}$ and $\arg\max_\ell \{Z_{t-\ell}\}$ otherwise. That is, the chain-rewinder seeks to maximize the observed variable, and once in every τ steps, it rewinds to a previous state with the highest observable.

An important subclass of strategies which we study in this work are *non-adaptive* rewinding strategies, where $A(Z_0, \ldots, Z_t) = A'(t)$, i.e., the rewinding strategy is a fixed function of the length and does not depend on the actual observations. Note that resettable strategies are special cases of non-adaptive strategies.

The main property of Markov chains we explore is the *identifiability* of the initial state. Specifically, given a p.o. Markov chain M and two candidate initial states $a, b \in \Omega$, we ask how long a rewinding strategy has to run (or how many "queries" it must make to M) to distinguish between these two starting states. The following definition formalizes this notion. It only considers identifiability from an information-theoretical point of view, i.e. the query complexity:

▶ **Definition 3.2** (Query Complexity). *Given a p.o. Markov chain M and rewinding strategy A, let $Z_0^a, Z_1^a, Z_2^a, \ldots$ denote the Markov chain with rewinding obtained from the initial state $X_0 = a$ and let $Z_0^b, Z_1^b, Z_2^b, \ldots$ correspond to the case $X_0 = b$. We say that (M, A, a, b) are (ε, T)-identifiable if $d_{\mathrm{TV}}((Z_0^a, \ldots, Z_T^a), (Z_0^b, \ldots, Z_T^b)) \geq \varepsilon$. The* query complexity *of distinguishing a from b in chain M with rewinding strategy A, denoted $\mathsf{QC}_M^A(a, b)$, is the minimum T such that (M, A, a, b) are $(1/3, T)$-identifiable. Having this, we define:*

- *the (adaptive) query complexity of distinguishing a from b to be:*

$$\mathsf{QC}_M(a,b) = \min_A \mathsf{QC}_M^A(a,b),$$

- *and define the analogous notion for non-adaptive rewinding strategies. Namely,*

$$\mathsf{NAQC}_M(a,b) = \min_{\text{non-adaptive } A} \mathsf{QC}_M^A(a,b).$$

We also consider identifiability from a computational standpoint. That is, we study the time complexity for algorithms that identify the initial state. Such an algorithm produces the rewinding strategy and queries the Markov chain accordingly. Then, it identifies the initial state based on all the observations. We call such algorithms *state identification algorithms*.

▶ **Definition 3.3** (State Identification Algorithms). *The input of a state identification algorithm consists of a p.o. Markov chain $M = (\Omega, P, O)$, two candidate initial states $a, b \in \Omega$, and a hidden initial state $X_0 \in \{a, b\}$. An (adaptive) state identification algorithm in every step $t \geq 0$, computes the rewinding strategy $A(Z_0, \ldots, Z_t)$, after which X_{t+1} is sampled according to $P_{X|Y=X_{t'}}$ with $t' = \max\{0, t - A(Z_0, \ldots, Z_t)\}$, and the observation $Z_{t+1} = O(X_{t+1})$ is revealed to the algorithm. A non-adaptive algorithm computes the rewinding strategy before sampling any states. The algorithm decides the number of steps T, and eventually outputs the identified initial state (a or b). We say the algorithm successfully identifies the initial state if the output is correct with probability $2/3$.*

Finally, we define a special subclass that we refer to as *canonical* p.o. Markov chains. While general p.o. Markov chains are of interest, we state our results in terms of canonical p.o. Markov chains for simplicity. We show later, in Theorem 6.2, that canonical chains capture all chains in our context.

▶ **Definition 3.4** (Canonical p.o. Markov Chains). *We say that p.o. Markov chain $M = (\Omega, P, O)$ is a* canonical *p.o. Markov chain if there exists a state $s \in \Omega$ that is a sink state (i.e., $P(s,s) = 1$ and equivalently $P(s,y) = 0$ for all $y \neq s$), and $O(a) = 1$ if $a = s$ and 0 otherwise.*

Thus, in a canonical p.o. Markov chain, the only observable signal is whether the chain has reached the sink or not; and once one reaches the sink, no further observations reveal information. Such chains are the hardest to learn and hence our focus.

3.1 Partitions

In the rest of this section, we introduce the notion of state partitions, i.e. partitions of Ω, a key tool both for designing rewinding strategies to distinguish between states, and proving lower bounds on the query complexity of this task. Throughout the section we use n as a shorthand for $|\Omega|$, $p(a,b)$ for $P(a,b) = \Pr(X_{t+1} = b \mid X_t = a)$, and $p(a,C)$ for $\sum_{b \in C} p(a,b)$. We use s to denote the sink state.

▶ **Definition 3.5** (Partition). *Given a set Ω, a partition $\mathcal{P}$ is a collection of sets such that each element $a \in \Omega$ appears in exactly one set of the partition. We often refer to these sets as classes, and use $\mathcal{P}(a)$ to denote the class of a in $\mathcal{P}$.*

▶ **Definition 3.6** (Total Variation Distance w.r.t. Partitions). *For two states a and b, and a partition $\mathcal{P}$, the total variation distance of $P(a,\cdot)$ and $P(b,\cdot)$ w.r.t. to $\mathcal{P}$ is*

$$d_{TV}^{\mathcal{P}}(a,b) := \frac{1}{2} \sum_{C \in \mathcal{P}} |p(a,C) - p(b,C)| = \max_{A \subseteq \mathcal{P}} \sum_{C \in A} p(b,C) - p(a,C).$$

Here, A is a collection of the classes in $\mathcal{P}$.

▶ **Lemma 3.7.** *Let $\mathcal{P}$ be a partition of Ω. Given two states a and b, and oracle access to $\mathcal{P}(x)$ for any state $x \in \Omega$, there exists a non-adaptive rewinding strategy that successfully distinguishes between initial states a and b with probability $1 - \varepsilon$ using*

$$O\left(\frac{\log 1/\varepsilon}{\left(d_{\mathrm{TV}}^{\mathcal{P}}(a,b)\right)^2}\right)$$

queries.

Proof. Let $d = d_{\mathrm{TV}}^{\mathcal{P}}(a,b)$, and let $A \subseteq \mathcal{P}$ be the collection of classes where the difference in the distributions is maximized:

$$A := \arg\max_{A \subseteq \mathcal{P}} \sum_{C \in A} p(b,C) - p(a,C).$$

Note that by definition $p(b,A) - p(a,A) = d$. Given a state $x \in \{a,b\}$, we draw $\frac{2\log 1/\varepsilon}{d^2}$ samples of the next step of x (according to distribution $P(x,\cdot)$), and let ρ be the fraction of them that land in A. If the state is a, the expected value of ρ is $p(a,A)$; if the state is b, the expected value is $p(b,A)$; and the difference between the two is equal to d. We declare that the state is a if $\rho < p(a,A) + \frac{d}{2}$ and that it is b otherwise. The probability of error when $x = a$ is equal to:

$$\Pr\left(\rho \geq p(a,A) + \frac{d}{2}\right) \leq \exp\left(-2 \cdot \frac{2\log 1/\varepsilon}{d^2} \cdot \left(\frac{d}{2}\right)^2\right) \leq \varepsilon.$$

Similarly, the probability of error when $x = b$ is equal to:

$$\Pr\left(\rho < p(a,A) + \frac{d}{2}\right) = \Pr\left(\rho < p(b,A) - \frac{d}{2}\right) \leq \varepsilon.$$ ◀

4 A Polynomially Optimal Non-Adaptive Algorithm

4.1 A Non-Adaptive Algorithm with Polynomial Queries

In this section, we present a non-adaptive algorithm that distinguishes any two (distinguishable) states a and b of an n-state canonical Markov chain $M = (\Omega, P)$ (see Definition 3.4) using $O_n(1) \cdot \mathsf{QC}_M(a,b)^{O(n)}$ queries, where recall $\mathsf{QC}_M(a,b)$ is the optimal query complexity of distinguishing a from b with a possibly adaptive rewinding strategy, and $O_n(1)$ is used to suppress terms dependent only on n. In particular, this implies that the gap between adaptive and non-adaptive query strategies is only polynomial so long as the number of states of the Markov chain is constant. We later show in Section 5 that this is the best one can hope for: there are choices of M, a, b where any non-adaptive rewinding strategy requires $\mathsf{QC}_M(a,b)^{\Omega(n)}$ queries.

Formally, we show:

▶ **Theorem 4.1.** *Let $M = (\Omega, P)$ be a canonical p.o. Markov chain with $n = |\Omega|$ states. There is a non-adaptive algorithm (Algorithm 1) that distinguishes any distinguishable states $a, b \in \Omega$ in time*

$$O_n(1) \cdot (\mathsf{QC}_M(a,b) \log \mathsf{QC}_M(a,b))^{2(n-2)} = O_n(1) \cdot \mathsf{QC}_M(a,b)^{O(n)}.$$

This is equivalent to Theorem 1.1.

4.2 An Informal Overview of the Algorithm

The algorithm defines a weighted directed graph $G_M = (V, E)$ where each vertex of the graph corresponds to a partitioning of Ω, the state space of the Markov chain. There is an edge from $\mathcal{P} \in V$ to $\mathcal{P}' \in V$ iff $\mathcal{P}'$ is a refinement of $\mathcal{P}$. Informally speaking, it would be helpful to view each partitioning $\mathcal{P}$ as a *test* that reveals which class of $\mathcal{P}$ a hidden state belongs to. With this view, we define the weight $w(\mathcal{P}, \mathcal{P}')$ of the directed edge $(\mathcal{P}, \mathcal{P}') \in E$ such that it upper bounds the cost of solving test $\mathcal{P}'$ having free access to test $\mathcal{P}$. Once we define this graph G_M, we compute the shortest path tree on G_M (e.g. using Dijkstra's algorithm) starting from the trivial partitioning $\mathcal{P}_0 = \{\Omega \setminus \{s\}, \{s\}\}$ that only separates the sink from the other states. Let $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k$ be the unique path in the tree from $\mathcal{P}_0$ to a partitioning $\mathcal{P}_k$. We will define the cost of $\mathcal{P}_k$ as

$$c(\mathcal{P}_k) = \prod_{i=0}^{k-1} w(\mathcal{P}_i, \mathcal{P}_{i+1}).$$

We prove that each test $\mathcal{P}$ can indeed be solved with (slightly more than) $c(\mathcal{P})$ queries. To distinguish state a from b, the algorithm takes the partitioning $\mathcal{P}$ that separates a from b and has the minimum cost $c(\mathcal{P})$. Our analysis shows that $c(\mathcal{P})$ can be upper bounded by $O_n(1) \cdot \mathsf{QC}_M(a, b)^{O(n)}$.

4.3 The Formal Argument

▶ **Definition 4.2** (Source Partition). *The **source partition** $\mathcal{P}_0$ is the trivial partition where the sink is a singleton class and all the remaining states form another class, i.e.* $\mathcal{P}_0 = \{\Omega \setminus \{s\}, \{s\}\}$.

For the purposes of this section, we only consider the partitions that separate s from every other state, i.e. partitions that refine $\mathcal{P}_0$.

▶ **Definition 4.3** (Partition Graph). *Given a canonical Markov chain $M = (\Omega, P)$, let the partition graph G_M be a weighted directed graph where each node is a partition of the states Ω. For a partition $\mathcal{P}_1$ and a refinement $\mathcal{P}_2$, there is a directed edge from $\mathcal{P}_1$ to $\mathcal{P}_2$ with weight*

$$w(\mathcal{P}_1, \mathcal{P}_2) = \max_{\substack{a,b: \\ \mathcal{P}_1(a)=\mathcal{P}_1(b) \\ \mathcal{P}_2(a)\neq\mathcal{P}_2(b)}} 1/\left(d_{\mathrm{TV}}^{\mathcal{P}_1}(a, b)\right)^2.$$

The lengths of the paths are defined multiplicatively. That is, the length of a path is the product of the weights of its edges. For a partition $\mathcal{P}$, its cost $c(\mathcal{P})$ is equal to the length of the shortest path from $\mathcal{P}_0$ to $\mathcal{P}$.

Below is a formal overview of the algorithm. The details of using the shortest paths to distinguish between states are given in Lemma 4.8. We use the notion of trees for rewinding. That is, if X_t is sampled by rewinding to $X_{t'}$ (i.e. according to distribution $P_{X|Y=X_{t'}}$), then $X_{t'}$ is the parent of X_t in the tree.

To analyze the query complexity of Algorithm 1, we show there exists a path of length $O_n(1) \cdot \mathsf{QC}(a, b)^{2(n-2)}$ to a partition that separates a and b (Lemma 4.7). Then, we show how this path can be used to distinguish between a and b (Lemma 4.8).

First, we prove some auxiliary claims. The following lemma essentially upper-bounds the weight of the smallest outgoing edge of a partition $\mathcal{P}$, based on two states a and b with $\mathcal{P}(a) = \mathcal{P}(b)$.

Algorithm 1 A non-adaptive algorithm with polynomial queries.

Input: A Markov chain $M = (\Omega, P)$, and two states a and b
Let G_M be the partition graph of the Markov chain as in Definition 4.3.
Compute the (multiplicative) shortest path tree on G_M rooted at the trivial partition $\mathcal{P}_0$.
Let $\mathcal{P}$ be a partition separating a and b that minimizes $c(\mathcal{P})$.
Let $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k = \mathcal{P}$ be the shortest path to $\mathcal{P}$.
Let $\varepsilon = \Theta_n\left(\frac{1}{c(\mathcal{P}) \log c(\mathcal{P})}\right)$.
Query a tree of height k, where the vertices at height i have degree $\Theta_n(\log(1/\varepsilon) \cdot w(\mathcal{P}_{i-1}, \mathcal{P}_i))$.
Use the observations in the tree to identify the class of each vertex at height i in $\mathcal{P}_i$ recursively (see Lemma 4.8 for the details).

▶ **Lemma 4.4.** *Take a partition $\mathcal{P}$ and any two states $a, b \in \Omega$ such that $\mathcal{P}(a) = \mathcal{P}(b)$. Then, there exists another partition $\mathcal{P}'$ that refines $\mathcal{P}$ and*

$$w(\mathcal{P}, \mathcal{P}') \leq \left(\frac{n-1}{d_{\mathrm{TV}}^{\mathcal{P}}(a,b)}\right)^2.$$

▶ **Corollary 4.5.** *For a partition $\mathcal{P}$, if $\mathcal{P}$ has no outgoing edge of weight at most h in the partition graph, then for any two states a and b in the same class of $\mathcal{P}$, it holds that*

$$d_{\mathrm{TV}}^{\mathcal{P}}(a,b) < \frac{n-1}{\sqrt{h}}.$$

Proof. Follows directly from Lemma 4.4. ◀

The following claim states that if there exists a partition $\mathcal{P}$ such that every two states in the same class are closer than θ w.r.t. $d_{\mathrm{TV}}^{\mathcal{P}}$, then distinguishing any two states in the same class of $\mathcal{P}$ is difficult (roughly speaking, requires $1/\theta$ queries).

▷ Claim 4.6. For a partition $\mathcal{P}$ that does not separate a and b, let

$$\theta(\mathcal{P}) \coloneqq \max_{\substack{x,y: \\ \mathcal{P}(x)=\mathcal{P}(y)}} d_{\mathrm{TV}}^{\mathcal{P}}(x,y).$$

Then, any (possibly adaptive) rewinding strategy that uses Q queries, can successfully distinguish between a and b with probability at most $\frac{1}{2} + \frac{Q \cdot \theta(\mathcal{P})}{2}$.

Combining Corollary 4.5 and Claim 4.6, for a partition $\mathcal{P}$ with $\mathcal{P}(a) = \mathcal{P}(b)$, we can upper-bound the smallest outgoing edge of $\mathcal{P}$, and prove there exists a short path from $\mathcal{P}_0$ to a partition that separates a and b.

▶ **Lemma 4.7.** *There exists a path of length $O_n(1) \cdot \mathsf{QC}(a,b)^{2(n-2)}$ from $\mathcal{P}_0$ to a partition that separates a and b.*

The following lemma describes how a path of length Q can be used to distinguish the states with (slightly more than) Q queries.

▶ **Lemma 4.8.** *Given a path $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k$ of length Q, one can identify the class of the initial state in $\mathcal{P}_k$ with $O_n(1) \cdot Q \log^k Q$ queries,[3] using a non-adaptive tree of height k where*

[3] recall $k \leq n-2$

on the i-th level the degree of every node is chosen proportionally to $w(\mathcal{P}_{i-1}, \mathcal{P}_i)$ and the class of each node on that level is determined w.r.t. $\mathcal{P}_i$.

Finally, we put Lemmas 4.7 and 4.8 together to derive Theorem 4.1.

Proof of Theorem 4.1. By Lemma 4.7, for any two states a and b, there exists a path of length at most $\mathsf{QC}(a,b)^{2(n-2)}$ ending at a partition that separates a and b. Therefore, the partition $\mathcal{P}$ obtained by computing shortest paths in step 4 of the algorithm has cost at most $\mathsf{QC}(a,b)^{2(n-2)}$. Lemma 4.8 shows how the shortest path to $\mathcal{P}$ can be used in steps 7–8, to identify the class of the initial state in $\mathcal{P}$ using $O_n(1) \cdot (\mathsf{QC}(a,b) \log \mathsf{QC}(a,b))^{2(n-2)}$ queries, hence distinguishing between a and b.

Regarding the runtime, observe that the strategy is implicitly (non-adaptively) computed in $O_n(1)$. The number of state partitions (number of vertices in G_M) is at most $n^n = O_n(1)$. For an edge $\mathcal{P} \to \mathcal{P}'$, the weight can be computed in time $O(n^3)$ by iterating over every relevant pair of states and computing the distance $d_{\mathrm{TV}}^{\mathcal{P}}$ in $O(n)$. Then the shortest path tree can be computed using a polynomial-time algorithm (e.g. Dijkstra's). After the shortest path is computed, the degree on each level of the query tree is determined. For each node of height i, its calls in $\mathcal{P}_i$ can be determined in time $O(2n^2 \cdot \log(1/\varepsilon) \cdot w(\mathcal{P}_{k-1}, \mathcal{P}_k))$, i.e. proportional to the degree. Therefore, the run time of the algorithm is $O_n(1) \cdot (\mathsf{QC}(a,b) \log \mathsf{QC}(a,b))^{2(n-2)} = O_n(1) \cdot (\mathsf{QC}(a,b))^{O(n)}$. ◀

5 The Polynomial Gap of Adaptivity

In this section, we prove the following theorem which directly implies Theorem 1.2.

▶ **Theorem 5.1.** *For any parameters $n, d \geq 1$, there exists a canonical Markov chain $M = (\Omega, P)$ with $n = |\Omega|$ states and two states $a, b \in \Omega$ such that*

$$\mathsf{QC}_M(a,b) = O(n^2 d), \qquad \mathsf{NAQC}_M(a,b) = \Omega(d^{(1-o(1))n}).$$

We remark that the optimal rewinding strategy, which distinguishes between a and b using $O(n^2 d)$ queries, can be trivially turned into an algorithm with the same run time. Therefore, Theorem 5.1 implies the same gap between the optimal adaptive and non-adaptive run times of state identification algorithms. As such, in this section, we abuse the notation and use the terms rewinding strategy and algorithm interchangeably.

We prove the claim given $d > 0$ and, the Markov chain M appeared in Figure 2.4. In this Markov chain, the states form a directed path toward the sink state, with states labeled as $q_1, ..., q_{n-2}$ from left to right. We label the last state as D. For all $i \in [n-3]$, state q_i goes to q_{i+1} with probability $\frac{1}{2d}$. The state q_{n-2} goes to the sink state s with the same probability. Each state in $q_1, ..., q_{n-2}$ goes to itself with probability $\left(1 - \frac{1}{d}\right)/2$. All $q_1, ..., q_{n-2}$ go to the state D with probability $\frac{1}{2}$, while state D goes to the sink with probability 1.

We prove that there exists an adaptive algorithm distinguishing q_1 and q_2 with $O(n^2 d)$ queries, while any non-adaptive algorithm needs at least $\Omega(d^{(1-o(1))n})$ queries. The intuition is as follows. An adaptive algorithm can easily test whether each new state opened is the D state or not by simply doing a test with constant queries, and opening another child instead to continue its path. This is not possible for non-adaptive algorithms that come across the D state with probability $\frac{1}{2}$, and when this happens the algorithm intuitively loses any information on the state it started from. So it needs a lot of paths to account for this loss of information.

▶ **Lemma 5.2.** *There exists an adaptive algorithm that distinguishes q_1 and q_2 in Markov chain M of Figure 2.4 with $O(n^2 d)$ queries.*

▶ **Lemma 5.3.** *Any non-adaptive algorithm distinguishing q_1 and q_2 in Markov chain M of Figure 2.4 needs $\Omega(d^{(1-o(1))n})$ queries.*

Proof of Theorem 5.1. Follows from combining Lemmas 5.2 and 5.3. ◀

6 Generality of Canonical Markov Chains

In this section, we show that identifying the initial state of a canonical Markov chain (Definition 3.4) is (essentially) as hard as the same task in any partially observable Markov chain via a reduction. That takes into account both the query complexity of the rewinding strategy and the runtime of the algorithm. For a state identification algorithm $\mathcal{A}$ we write $\mathrm{T}(\mathcal{A})$ for the time it spends outside the oracle that samples the next state.

In the generalized version, a Markov chain $M = (\Omega, P)$ is complemented by an observation function $O : \Omega \to \Sigma$, i.e. every state $x \in \Omega$ in the Markov chain outputs an observation $O(x) \in \Sigma$. The states remain hidden, but in every step, the algorithm can see the observation and use it to identify the initial states. A canonical Markov chain can be expressed in this generalized version by letting the set of possible observations be $\Sigma = \{0, 1\}$, and letting $O(x) = \mathbf{1}(x = s)$ where s is the sink state. That is, the observation made at any state is whether it is the sink state. First, We show that any adaptive strategy taking Q queries in the generalized model can be emulated in the canonical Markov chain using $O(|\Sigma|Q)$ queries. Secondly, we show that for non-adaptive algorithms, the gap between the query complexity of canonical and generalized Markov chains is at most near-quadratic.

Formally, we define a reduction from a generalized Markov chain M to a canonical Markov chain $\hat{M}$ below. See Figures 6.1 and 6.2 for an example of this reduction.

▶ **Definition 6.1** (Reduction to Canonical Markov chains)**.** *Let $M = (\Omega, P)$ be a Markov chain with state space Ω, transition matrix P, and an observation function $O : \Omega \to \Sigma$ that maps each state to an observable output. Let the emulated Markov chain of M be a canonical Markov chain $\hat{M} = (\hat{\Omega}, \hat{P})$ with state space $\hat{\Omega}$ and transition matrix $\hat{P}$ as follows:*

For each state $x \in \Omega$, there exists a unique corresponding state $\hat{x} = \phi(x)$ in $\hat{\Omega}$, where $\phi : \Omega \to \hat{\Omega}$ is an injective mapping. Let $k = |\Sigma|$ and and $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$. Given a parameter $0 < q < 1$ we have

1. *For every pair of states $x, x' \in \Omega$ with transition probability $p_{x,x'}$ in P, we introduce an extended path in $\hat{M}$ consisting of a series of intermediate states $d_1^{x,x'}, d_2^{x,x'}, \ldots, d_{k-1}^{x,x'}$ with transition probability 1 between consecutive states, such that $\hat{x}$ transitions to $\hat{x}'$ via these intermediate states. $\hat{x}$ transitions to $d_1^{x,x'}$ with probability $qp_{x,x'}$ and $d_{k-1}^{x,x'}$ transitions to $\hat{x}'$ with probability 1.*
2. *To emulate the observation function O, we introduce a set of **special states** $\sigma_1, \sigma_2, \ldots, \sigma_k = s$ in $\hat{M}$, each representing a unique observation in Σ. s is the sink state of the canonical Markov chain $\hat{M}$. Each state $\hat{x} \in \hat{\Omega}$ in $\hat{M}$ transitions to a special state corresponding to its observation in M with probability $1 - q$. Special state σ_i transitions to σ_{i+1} with probability 1 for $i \in [k - 1]$.*

▶ **Theorem 6.2.** *Let $M = (\Omega, P)$ be a Markov chain in the generalized model with an observation function $O : \Omega \to \Sigma$, Then, there exists a canonical Markov chain $\hat{M} = (\hat{\Omega}, \hat{P})$ states and an injective mapping of the states $\phi : \Omega \to \hat{\Omega}$, such that for any two states $a, a' \in \Omega$,*

$$\mathsf{QC}_{\hat{M}}(\phi(a), \phi(a')) = \Theta\big(|\Sigma|\mathsf{QC}_M(a, a')\big).$$

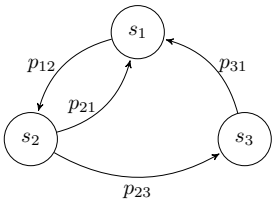

Figure 6.1 Represents a general Markov chain M with observations $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$, where $O_{s_1} = \sigma_1, O_{s_2} = \sigma_2$, and $O_{s_3} = \sigma_2$.

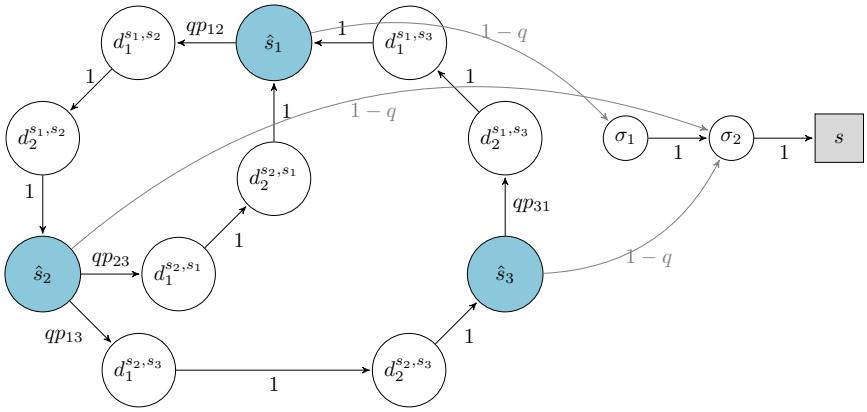

Figure 6.2 Represents a canonical Markov chain $\hat{M}$, with parameter $0 < q < 1$, where there is an injection ϕ that reduces distinguishing states a and a' in M to distinguishing $\phi(a)$ and $\phi(a')$ in $\hat{M}$.

▶ Remark 6.3. This reduction preserves running time up to the same multiplicative overhead: for any adaptive state identification algorithm $\mathcal{A}$ for M that makes Q queries and runs in time $Time(\mathcal{A})$, there is an algorithm $\hat{\mathcal{A}}$ for $\hat{M}$ that runs in time $Time(\mathcal{A}) + O(kQ)$, where $k = |\Sigma|$. Also, any adaptive algorithm $\hat{\mathcal{A}}$ for $\hat{M}$ with Q' queries and time $Time(\hat{\mathcal{A}})$ yields an algorithm for M with $\Omega(Q'/k)$ queries and time $Time(\hat{\mathcal{A}}) + O(Q')$.

▶ **Theorem 6.4.** *Let $M = (\Omega, P)$ be a Markov chain in the generalized model with an observation function $O : \Omega \to \Sigma$, Then, there exists a canonical Markov chain $\hat{M} = (\hat{\Omega}, \hat{P})$ states and an injective mapping of the states $\phi : \Omega \to \hat{\Omega}$, such that for any two states $a, a' \in \Omega$,*

$$\Omega\big(|\Sigma|\mathsf{NAQC}_M(a,a')\big) \leq \mathsf{NAQC}_{\hat{M}}(\phi(a),\phi(a')) \leq \tilde{O}\big(|\Sigma|\mathsf{NAQC}_M(a,a')^2\big).$$

▶ Remark 6.5. If a non-adaptive algorithm $\mathcal{A}$ for Markov chain M uses Q queries and runs in time $Time$, then the induced non-adaptive algorithm $\widehat{\mathcal{A}}$ for $\hat{M}$ can be implemented with $\tilde{O}(kQ^2)$ queries and total running time $Time(\mathcal{A}) + \tilde{O}(kQ^2)$, where $k = |\Sigma|$. Also, any non-adaptive $\hat{\mathcal{A}}$ for $\hat{M}$ with Q' queries and time $Time(\hat{\mathcal{A}})$ yields a non-adaptive algorithm for M with $\Omega(Q'/k)$ queries and running time $Time(\hat{\mathcal{A}}) + O(Q')$.

References

1 David J. Aldous and Umesh V. Vazirani. "Go with the winner" algorithms. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 492–501. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365742.

2 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786. SIAM, 2019. doi:10.1137/1.9781611975482.48.

3 Sepehr Assadi and Chen Wang. Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 10:1–10:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.10.

4 Yossi Azar, Andrei Z. Broder, Anna R. Karlin, Nathan Linial, and Steven Phillips. Biased random walks. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 1–9, New York, NY, USA, 1992. Association for Computing Machinery. doi:10.1145/129712.129713.

5 Amir Azarmehr, Soheil Behnezhad, Alma Ghafari, and Madhu Sudan. Lower bounds for non-adaptive local computation algorithms. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, 2025. doi:10.1109/FOCS63196.2025.00078.

6 Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 873–884. IEEE, 2021. doi:10.1109/FOCS52979.2021.00089.

7 Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. Local computation algorithms for maximum matching: New lower bounds. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 2322–2335, 2023. doi:10.1109/FOCS57990.2023.00143.

8 Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. Sublinear time algorithms and complexity of approximate maximum matching. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 267–280, 2023. doi:10.1145/3564246.3585231.

9 Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. Approximating maximum matching requires almost quadratic time. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 444–454, 2024. doi:10.1145/3618260.3649785.

10 Soheil Behnezhad, Mohammad Roghani, Aviad Rubinstein, and Amin Saberi. Sublinear algorithms for TSP via path covers. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 19:1–19:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.19.

11 Michael A Bender and Dana Ron. Testing properties of directed graphs: acyclicity and connectivity. *Random Structures & Algorithms*, 20(2):184–205, 2002. doi:10.1002/RSA.10023.

12 Sayan Bhattacharya, Peter Kiss, and Thatchaphol Saranurak. Dynamic $(1+\varepsilon)$-approximate matching size in truly sublinear update time. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1563–1588. IEEE, 2023. doi:10.1109/FOCS57990.2023.00095.

13 Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 235–244. ACM, 2000. doi:10.1145/335305.335334.

14 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2001. doi:10.1007/3-540-48224-5_16.

15 Yu Chen, Sanjeev Khanna, and Zihan Tan. Query complexity of the metric steiner tree problem. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 4893–4935. SIAM, 2023. doi:10.1137/1.9781611977554.CH179.

16 Yu Chen, Sanjeev Khanna, and Zihan Tan. Sublinear algorithms and lower bounds for estimating MST and TSP cost in general metrics. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 37:1–37:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.37.

17 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 175–183. ACM, 2004. doi:10.1145/1007352.1007386.

18 Ioana Dumitriu, Prasad Tetali, and Peter Winkler. On playing golf with two balls. *SIAM J. Discret. Math.*, 16(4):604–615, 2003. doi:10.1137/S0895480102408341.

19 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 406–415. ACM, 1997. doi:10.1145/258533.258627.

20 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. doi:10.1137/0218012.

21 Svante Janson and Yuval Peres. Hitting times for random walks with restarts. *SIAM J. Discret. Math.*, 26(2):537–547, 2012. doi:10.1137/100796352.

22 Slobodan Mitrović, Ronitt Rubinfeld, and Mihir Singhal. Locally Computing Edge Orientations. In *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 89:1–89:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.89.

23 Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.81.

24 Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 410–414. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814612.

25 Vihan Shah. Sublinear-time lower bounds for approximating matching size using non-adaptive queries. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5027–5065. SIAM, 2026. doi:10.1137/1.9781611978971.182.

26 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 225–234. ACM, 2009. doi:10.1145/1536414.1536447.

A Further Connections to Sublinear-Time Graph Algorithms

In this section, we further elaborate on the connections to sublinear-time graph algorithms by giving a concrete example where the lower bound is captured by Markov chains with rewinding.

We examine the lower bound of [11] for testing acyclicity in directed graphs. Given adjacency list access, the algorithm must determine whether the input graph is acyclic or ε-far from acyclic (i.e., at least an ε-fraction of the edges must be removed so that no directed cycles remain in the graph) with a probability of $\frac{2}{3}$. They prove that, for $\varepsilon \leq \frac{1}{16}$, any such algorithm requires at least $\Omega(n^{1/3})$ queries to the adjacency list. We show that this lower bound is captured by our model. First, we examine the case where the algorithm has access

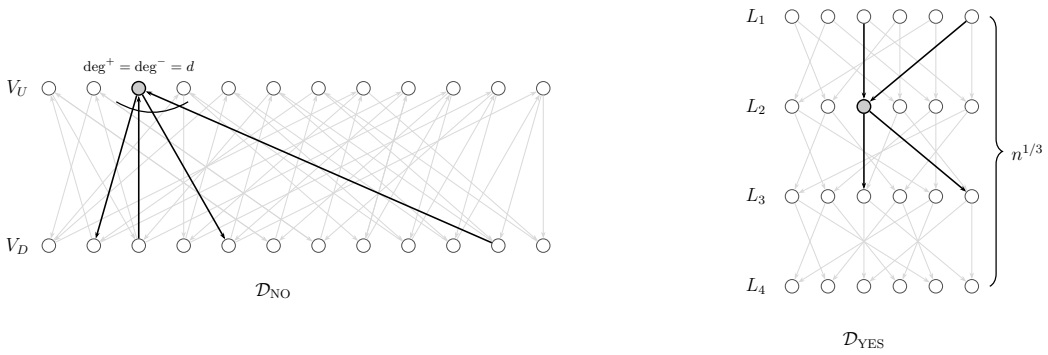

Figure A.1 Sample graphs from the $\mathcal{D}_{\text{YES}}$ (acyclic) and $\mathcal{D}_{\text{NO}}$ (far from acyclic) distributions. The typical vertex has an in-degree and out-degree of d.

only to the outgoing edges (and the out-degree) of a vertex, and later extend it to a setting where the algorithm has access to both the outgoing and incoming adjacency lists at the same time (as well as the in-degree and out-degree).

We start by reviewing the construction of [11]. By Yao's minimax lemma, to prove the lower bound, it suffices to construct two input distributions $\mathcal{D}_{\text{YES}}$ and $\mathcal{D}_{\text{NO}}$, respectively consisting of acyclic and far-from-acyclic graphs, such that any deterministic algorithm with $O(n^{1/3})$ queries fails to distinguish between them with a sufficiently large probability. The construction uses a constant degree-parameter $d \geq 128$ shared between $\mathcal{D}_{\text{YES}}$ and $\mathcal{D}_{\text{NO}}$.

The $\mathcal{D}_{\text{NO}}$ distribution constructs a graph by randomly partitioning the vertices into two groups, V_U and V_D, each of size $n/2$. For the edges, a bipartite d-regular graph between V_U and V_D is chosen uniformly at random and directed towards V_D, and another one is chosen similarly and directed towards V_U. As a result, each vertex in the graph has both in-degree and out-degree equal to d, and the graph is ε-far from acyclic with a high probability of $1 - 2^{-n}$, when $\varepsilon < \frac{1}{16}$ and $d \geq 128$ [11, Lemma 5].

The $\mathcal{D}_{\text{YES}}$ distribution constructs a graph by randomly partitioning the vertices into $n^{1/3}$ groups, $L_1, L_2, \ldots, L_{n^{1/3}}$, each of size $n^{2/3}$. For the edges, for each $1 \leq i < n^{1/3}$, a d-regular graph between L_i and L_{i+1} is chosen uniformly at random and directed towards L_{i+1}. As a result, the graph has no directed cycles and every vertex, except those of L_1 and $L_{n^{1/3}}$, has an in-degree and out-degree equal to d. The distributions are depicted in Figure A.1.

Now, we present the Markov chain that captures this lower-bound construction. The Markov chain consists of two parts: one for $\mathcal{D}_{\text{NO}}$ and one for $\mathcal{D}_{\text{YES}}$. In the NO part, each of the vertex groups V_U and V_D is represented by a state, v_U and v_D respectively. Considering that in the graph, the outgoing edges of V_U go to V_D and the outgoing edges of V_D go to V_U, in the Markov chain, we let v_D transition to v_U, and let v_U transition to v_D with probability 1. To model starting at a random vertex, we use an initial state x_{NO} that transitions to v_D or v_U with probability $\frac{1}{2}$ each. Similarly, in the YES part, each vertex group L_i is represented by a state ℓ_i. Each state ℓ_i for $1 \leq i < n^{1/3}$, transitions to the ℓ_{i+1} with probability 1. To model starting at a random vertex, we use an initial state x_{YES} that transitions to each of $\ell_1, \ell_2 \ldots, \ell_{n^{1/3}}$ with probability $\frac{1}{n^{1/3}}$. As the algorithm interacts with the Markov chain, the only observation it makes is whether the drawn state is $\ell_{n^{1/3}}$ or not. The Markov chain is illustrated in Figure A.2.

Next, we elaborate on how this Markov chain captures the lower bound. The key idea is that an algorithm that makes $O(n^{1/3})$ queries, does not discover any vertex more than once, with a sufficiently high constant probability. That is, the set of edges discovered by the sublinear algorithm does not contain any cycles, even when the directions are ignored. Let $(u_1, i_1), \ldots, (u_q, i_q)$ be the set of queries made to the adjacency lists. We assume without

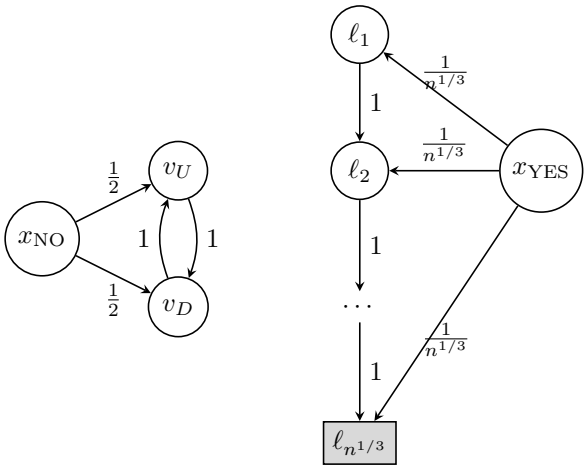

Figure A.2 A Markov chain that models exploring the input graph drawn from the $\mathcal{D}_{\text{YES}}$ and $\mathcal{D}_{\text{NO}}$ distributions. Distinguishing the initial states x_{YES} and x_{NO} is equivalent to distinguishing between the two input distributions.

loss of generality that the queries are unique (i.e., the same query is not made twice), and that the degree of each vertex is revealed to the algorithm as it is discovered.

▶ **Lemma A.1** ([11, Lemma 7]). *Consider any algorithm that makes $q \leq \frac{1}{4} \cdot n^{1/3}$ adjacency-list queries, and let $v_1, \ldots v_q$ be the answers, where $v_k \in V \cup \{\perp\}$. Then, with probability $1 - \frac{1}{16}$, no vertex appears in the answers more than once.*

As a result, we can assume without loss of generality that no vertex appears twice in the answers, since $\frac{1}{16}$ is a negligible probability and when a vertex appears twice, we can simply presume that the algorithm successfully distinguishes between the input distributions. Making this assumption, we argue that distinguishing between the input distributions $\mathcal{D}_{\text{YES}}$ and $\mathcal{D}_{\text{NO}}$ is equivalent to distinguishing between initial states x_{YES} and x_{NO} in the Markov chain. Since there are no repeated vertices, the subgraph explored by the algorithm is a set of rooted trees. When the algorithm explores the neighbors of a vertex u in a group of vertices A, the discovered neighbor v is a random vertex from a neighboring group B. The distribution of B is solely determined by A and does not depend on u.[4] Furthermore, the algorithm observes only the out-degree of each discovered vertex (which is different only for vertices in $L_{n^{1/3}}$), and not its group. This is paralleled by the transition probabilities in the Markov chain, and the partial observation of the state, which only differentiates $\ell_{n^{1/3}}$.

As a result, to prove the lower bound for acyclicity testing, it suffices to prove the following in the context of Markov chains:

▶ **Lemma A.2.** *Any algorithm that distinguishes between the initial states x_{YES} and x_{NO} with a probability of $5/8$, must make $\frac{1}{4} \cdot n^{1/3}$ queries to the Markov chain.*

The proof in [11] essentially argues the same thing, without ever explicitly defining a Markov chain. We give a high-level overview of their proof here. Since the only observation the algorithm can make is whether it has reached $\ell_{n^{1/3}}$ or not, it suffices to bound the probability of reaching $\ell_{n^{1/3}}$ when the initial state is x_{YES}. The next state drawn from x_{YES}

[4] in this construction, B is in fact a deterministic function of A.

is always a state ℓ_i, where $i \in \{1, 2, \ldots, n^{1/3}\}$ is uniformly random. As a result, exploring a tree of depth D from such a state, reaches $\ell_{n^{1/3}}$ with probability of at most $\frac{D+1}{n^{1/3}}$. Therefore, since the sum of the depths of the explored trees is $O(n^{1/3})$, the probability of reaching $\ell_{n^{1/3}}$ throughout the algorithm can be bounded by a constant.

Finally, we show that the lower bound is captured by our model in the more general case, where the algorithm is allowed to query both the incoming and outgoing adjacency lists. To do so, we use a simple gadget. On the YES side of the Markov chain, for each state ℓ_i, we create two paths of length two: A path $\ell_i \to d_i \to \ell_{i+1}$, corresponding to traversing the outgoing edges of L_i, and a path $\ell_i \to u_i \to \ell_{i-1}$, corresponding to traversing the incoming edges of L_i (the NO side is handled similarly). Additionally, we extend the observations made by the algorithm. Other than observing whether the current state is $\ell_{n^{1/3}}$, the algorithm can also observe whether the current state is (1) ℓ_1, the first layer corresponding to vertices with zero in-degree, (2) an auxiliary state d_i, indicating the traversal of an outgoing edge, and (3) an auxiliary state u_i, indicating the traversal of an incoming edge. This allows the algorithm to choose which of the directions it takes, while increasing the number of queries to the Markov chain only by a constant factor. See Section 6 for a discussion of Markov chains with a constant-sized observation alphabet.