

Testing Spreading Behavior in Networks with Arbitrary Topologies

Augusto Modanese  

Aalto University, Finland

Yuichi Yoshida  

National Institute of Informatics, Tokyo, Japan

Abstract

Given the full topology of a network, how hard is it to test if it is evolving according to a local rule or is far from doing so? Inspired by the works of Goldreich and Ron (J. ACM, 2017) and Nakar and Ron (ICALP, 2021), we initiate the study of property testing in dynamic environments with arbitrary topologies. Our focus is on the simplest non-trivial rule that can be tested, which corresponds to the 1-BP rule of bootstrap percolation and models a simple spreading behavior: Every “infected” node stays infected forever, and each “healthy” node becomes infected if and only if it has at least one infected neighbor. Our results are subdivided into two main groups:

- If we are testing a single time step of evolution, then the query complexity is $O(\Delta/\varepsilon)$ or $\tilde{O}(\sqrt{n}/\varepsilon)$ (whichever is smaller), where Δ and n are the maximum degree of a node and the number of vertices in the underlying graph, respectively. We also give lower bounds for both one- and two-sided error testers that match our upper bounds up to $\Delta = o(\sqrt{n})$ and $\Delta = O(n^{1/3})$, respectively. If ε is constant, then the first of these also holds against adaptive testers.
- When testing the environment over T time steps, we have two algorithms that need $O(\Delta^{T-1}/\varepsilon T)$ and $\tilde{O}(|E|/\varepsilon T)$ queries, respectively, where E is the set of edges of the underlying graph.

All of our algorithms are one-sided error, and all of them are also non-adaptive, with the single exception of the more complex $\tilde{O}(\sqrt{n}/\varepsilon)$ -query tester for the case $T = 2$.

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

Keywords and phrases Property testing, bootstrap percolation, local phenomena, expander graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.112

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version:* <https://arxiv.org/abs/2309.05442> [17]


Funding *Augusto Modanese:* Supported by the Helsinki Institute for Information Technology (HIIT). Part of this work done while affiliated with the Karlsruhe Institute of Technology (KIT) and visiting the NII as a JSPS International Research Fellow.

Yuichi Yoshida: Partly supported by JSPS KAKENHI Grant Number 18H05291 and 20H05965.

Acknowledgements We would like to thank Jukka Suomela for interesting discussions.

1 Introduction

Imagine we are observing the state of a network as it evolves over time. The network is static and we have complete knowledge about the connections; it is too large for us to keep track of the state of every single node, though nevertheless we are able to query nodes directly and learn their states. We might hypothesize that the global behavior can be explained by a certain local rule that is applied at every node, and we would like to verify if our hypothesis is correct or not. In this paper, we focus on this question: *How hard is it to test, given a local rule R , if the network is following R or is far from doing so?*

 © Augusto Modanese and Yuichi Yoshida;
licensed under Creative Commons License CC-BY 4.0
51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 112; pp. 112:1–112:20

 Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Following previous works [10, 18], we refer to the series of configurations assumed by the network over time as the *environment* ENV that we are observing. The network itself is static and its connections defined by a graph $G = (V, E)$. The local rule R is a map (admitting a finite description) from the states that a node observes in its neighborhood (including the node itself) to the new state it will assume in the next time step. Plausible scenarios that could be modeled in this context include not only rumor dissemination in social networks but also spreading of infectious diseases (where the connections between nodes represent proximity or contact between the organisms that we are observing). As is common in property testing [5], we assume that the bottleneck of this problem is keeping track of the states across the entire network, and hence we consider only the number of queries made by a testing algorithm as its measure of efficiency (and otherwise assume that the algorithm has access to unbounded computational resources).

1.1 Problem Setting

There exist two previous works [10, 18] that study the problem of determining whether ENV evolves according to R in the context of property testing (and, in the case of [10], also in the context of learning theory). In these works, the structure underlying ENV is always a cellular automaton (in the case of [18] one-dimensional, whereas [10] also considers automata of multiple dimensions), and thus ENV corresponds to the time-space diagram of such an automaton. This perspective is certainly meaningful when we are interested in phenomena that take place on a lattice or can be represented in such grid-like structures, for instance the movement of particles on a surface or across three-dimensional space. Nevertheless there are limits as to what can be modeled in this way. A prominent example are social networks, in which the connections hardly fit well into a regular lattice (even with several dimensions).

In this work we cast off these restraints and instead take the radically different approach of making no assumptions about the underlying structure or the space it is embedded in. Our only requirement is that it corresponds to a static graph G that is known to us in advance. This leaves a much broader avenue open when it comes to applications. In addition, the rule that we consider is effectively the simplest rule possible in such a setting that is not trivial. As we will see, despite the rule being very simple, it is rather challenging to fully determine the complexity of the problem. Indeed, compared to the previous works mentioned above, it might seem as if our progress is more modest; however, one should keep in mind that, in our case, the underlying network G has a much more rich structure (whereas in cellular automata we are dealing with a highly regular one).

The rule that we study is the 1-BP rule of *bootstrap percolation* [11, 20, 13, 2]. For $\tau \in \mathbb{N}_0$, the rule τ -BP is defined based on two states, *black* and *white*, as follows: If a node is black, then it always remains black; if a node is white, then it turns black if and only if it has at least τ black neighbors. These rules were originally inspired in the behavior observed in certain materials, and they are very naturally suited for modeling spreading phenomena.

Seen from the lenses of property testing, testing for the 1-BP rule in a sense resembles monotonicity testing [7]. Although we cannot directly apply one strategy to the other, if we view black as 1 and white as 0, then in both cases we have a violation whenever we see a 1 preceding a 0. The difference is that in 1-BP *every* 1 must arise from a preceding 1, whereas in the case of monotonicity we are happy if an isolated 0 spontaneously turns into a 1.

Another way of modeling the 1-BP rule is as a *constraint satisfaction problem* (CSP). CSPs have been studied in the context of property testing to some extent [4, 6]. We can characterize 1-BP by two constraints: A black node in step t implies every one of its neighbors is also black in step $t + 1$; meanwhile, a node is white in step $t + 1$ if and only if every one of

its neighbors in step t was white. Then we can recast testing if ENV follows 1-BP as testing if ENV is a satisfying assignment for these constraints. Nevertheless, although this seems to be a useful rephrasing of the problem, the current methods in CSPs in the context of property testing are not sufficient to tackle it. And, even if we could indeed test either constraint with a sublinear number of queries, ENV being close to satisfying both constraints would not necessarily imply that ENV is close to satisfying their intersection.

1.2 Results and Techniques

We now present our results and the methods used to obtain them. As this is a high-level discussion, formal definitions are postponed to Section 2, which the reader is invited to consult as needed.

The relevant parameters for the results are the number of nodes n in the graph $G = (V, E)$, the number of steps T during which the environment ENV evolves, the maximum degree Δ of G , and the accuracy parameter $\varepsilon > 0$. The size of the environment is nT , which is the baseline for linear complexity in this context (instead of n). We write $\text{ENV} \in \text{1-BP}$ to indicate that ENV follows the 1-BP rule and $\text{dist}(\text{ENV}, \text{1-BP}) \geq \varepsilon$ when it is ε -far from doing so, that is, one must flip at least εnT colors in ENV in order for 1-BP to be obeyed everywhere. (As already mentioned, see Section 2 for the precise definitions.)

In most cases we will be interested in optimizing the dependency of the query complexity on Δ . This is due to the fact that, intuitively, graphs with small Δ should be easier to verify locally, that is, by looking only at each node's neighborhood. (Indeed, this is the strategy followed by the first algorithm we present below in Theorem 1.)

Another desirable property that we wish our algorithms to have is *non-adaptiveness*; that is, the algorithm first produces a list of queries (without looking at the input), gathers their results, and then decides whether to accept or not. This is in contrast to an *adaptive* algorithm, which may perform later queries based on the answers it has seen so far. A third property “in-between” these is *time-conformability*, meaning that the algorithm does not make queries in step t if it has already queried nodes at some later step $t' > t$. It is easy to see that the existence of a non-adaptive algorithm with query complexity q implies a time-conforming algorithm with the same complexity: Just gather the q queries in a list, sort them according to the time step queried, and then execute the queries in order. The converse is not true in general, however, since a time-conforming algorithm might choose its queries in a later time step based on what it has seen beforehand (or in the same time step, even).

Recalling our motivation of testing the evolution of huge networks, we see that non-adaptive algorithms are the most desirable because the queries may all be performed in parallel (at each time step). In case this cannot be achieved, an adaptive, time-conforming algorithm is still satisfactory, even though it might require “freezing” the network at a specific time step (so that the algorithm has time to gather the results received and decide on the next queries to make on the same time step). Adaptive algorithms that violate time-conformability are not particularly desirable since they require “rewinding” the state of the network back in time. Nevertheless, depending on the nodes' capabilities, there might still be strategies to cope with this; for example, if T is small, it is plausible to have nodes can cache their state in previous steps (and thus answer any of the algorithm's queries, even about previous states).

In this paper, we study two different settings: testing a single time step of evolution ($T = 2$) and testing multiple steps ($T > 2$). In the first case we prove both upper and lower bounds, which also match up to certain values of Δ . In the second we show only upper bounds, but which suffice to demonstrate that the problem admits non-trivial testers, at least for moderate (non-constant) values of Δ .

1.2.1 The Case $T = 2$

Let us first discuss our results for the case where $T = 2$. In this case there is a natural graph-theoretical rephrasing of the problem: For $t \in \{1, 2\}$, let S_t be the set corresponding to $\text{ENV}(\cdot, t)$ where we see $\text{ENV}(\cdot, t)$ as an indicator function (i.e., S_t is exactly the set of nodes $v \in V$ for which $\text{ENV}(v, t) = 1$). Then $\text{ENV} \in 1\text{-BP}$ if and only if S_2 is dominated by S_1 (in graph-theoretic terms).¹ From this perspective, the distance from ENV to 1-BP is the (relative) total number of nodes we need to add or remove from either of S_1 or S_2 in order for the domination property to be satisfied.

The hardness of the problem in this case is highly dependent on the maximum degree Δ . Our first result is that there is a very natural and simple algorithm that achieves query complexity $O(\Delta/\varepsilon)$.

► **Theorem 1.** *Let $T = 2$ and $\varepsilon > 0$. There is a non-adaptive, one-sided error algorithm with query complexity $O(\Delta/\varepsilon)$ that decides whether $\text{ENV} \in 1\text{-BP}$ or $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$.*

The algorithm simply selects nodes at random and then queries their entire neighborhoods in both time steps. Since we are dealing with a local rule, this is sufficient to detect if ENV contains too many violations of the rule or not. One detail that needs care here is that, in general, our notion of distance does *not* match the number of violations of the rule. Nevertheless, as we show, the cases where it does not are only playing in our favor, and so this strategy always succeeds.

It turns out that this algorithm is optimal when we are in regimes where there is a constant $b \geq 2$ such that $\varepsilon = \Omega(\Delta^b/n)$. We also prove lower bounds for the case where $b \geq 1$, which are especially useful in regimes where Δ is larger than \sqrt{n} .

► **Theorem 2.** *There is a constant $\varepsilon_0 > 0$ such that the following holds: Let $\varepsilon = \Omega(\Delta^b/n)$ be given where $b \geq 1$ is constant, and let $\varepsilon \leq \varepsilon_0$. Then deciding if $\text{ENV} \in 1\text{-BP}$ or is ε -far from 1-BP with a one-sided error tester requires at least q queries in general, where:*

1. *If $b > 2$, then $q = \Omega(\Delta/\varepsilon)$ if the tester is non-adaptive or $q = \Omega(1/\varepsilon + \Delta)$ if it is adaptive.*
2. *If $b = 2$, then $q = \Omega(\Delta/\varepsilon \log \Delta)$ if the tester is non-adaptive or $q = \Omega(1/\varepsilon + \Delta/\log \Delta)$ if it is adaptive.*
3. *If $1 \leq b < 2$, then $q = \Omega(\Delta^{b-1}/\varepsilon)$ if the tester is non-adaptive or $q = \Omega(1/\varepsilon + \Delta^{b-1})$ if it is adaptive.*

Note the lower bounds hold even for adaptive testers in general, even for those that do not respect time-conformability.

If we are only interested in the regime where ε is constant, then setting $b = \log_{\Delta} n$ above we obtain a lower bound of $\Omega(\Delta)$ whenever $\Delta = O(n^{1/2-c})$ for a constant $c > 0$. This is matched by the upper bound of Theorem 1. For $\Delta = \Theta(\sqrt{n})$, the lower bound is $\Omega(\Delta/\log n)$; for larger Δ the lower bound becomes $\Omega(n/\Delta)$ and thus deteriorates as Δ increases.

The lower bound is based on an adequate construction of expander graphs. More specifically, the expanders we construct are bipartite, Δ -regular, and have *distinct* expansion guarantees for sets of nodes on either side. This is needed because the expansion in one direction guarantees ε -farness whereas the one in the other direction yields the actual lower bound on the number of queries that a correct algorithm must make.

¹ Technically the definition of domination is so that A dominates B if and only if $B \subseteq A \cup N(A)$. Using this definition the equivalence is only true if the graph G contains self-loops everywhere. (Nevertheless, adding self-loops everywhere does not impact the maximum degree, which is the relevant parameter here.) The equivalence is certainly true if we change the definition so that A dominates B if $B \subseteq N(A)$.

The hard instances themselves are simple: We color a moderately large set B of randomly chosen nodes black in the second time step and leave the rest colored white. The intuition is that, since B is chosen at random, it will not match nicely with a cover $C = \bigcup_{u \in S} N(u)$ induced by some set of nodes S in the first time step; that is, the symmetric difference between B and C will likely be large, giving us ε -farness. At the same time, since a one-sided error algorithm A cannot reject good instances, it is hard for it to detect that there is something wrong with B without having to “cover” a considerable number of nodes in either step. Indeed, in order to ascertain that a node $v \in B$ is incorrect, A must verify that there is no black node in $N(v)$ in the first step; if the existence of some black $u \in N(v)$ is compatible with its view, then there is no contradiction to v being black, and hence A cannot reject. Querying all of $N(v)$ requires $\Omega(\Delta)$ queries, but it is also possible for A to determine the colors *indirectly* by querying neighbors of nodes in $N(v)$. To obtain the lower bound we show that this other strategy also requires too many queries – although it might not be as inefficient when Δ is large (thus explaining why we get a weaker result in that case).

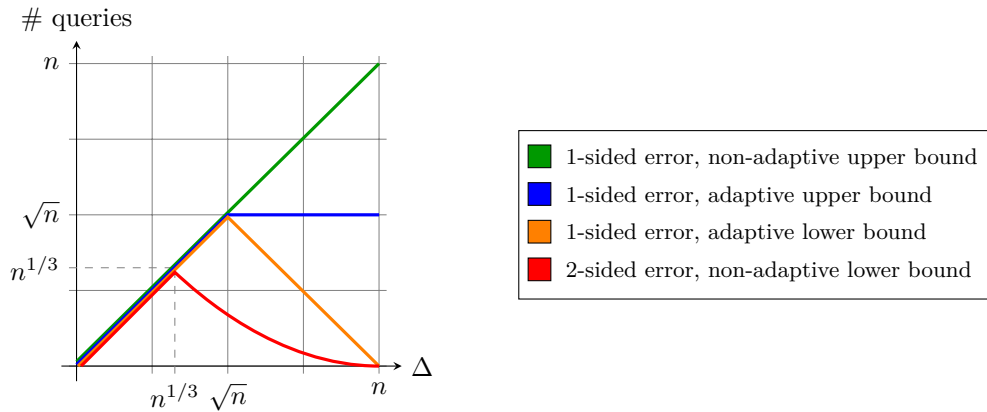
For two-sided error algorithms, we are able to prove similar, though slightly more modest lower bounds. These are also based on expander graphs but require a more complex set of instances for the argument to go through.

► **Theorem 3.** *There are constants $\varepsilon_0, \zeta > 0$ such that, for any $0 < \varepsilon \leq \varepsilon_0$ with $\varepsilon \geq \zeta \Delta^b/n$ where $b \geq 1$ is constant, deciding if $\text{ENV} \in \text{1-BP}$ or is ε -far from 1-BP with a non-adaptive, two-sided error tester requires q queries in general, where:*

- If $b > 3$, then $q = \Omega(\Delta/\varepsilon)$.
- If $b = 3$, then $q = \Omega(\Delta/\varepsilon \log \Delta)$.
- If $1 \leq b < 3$, then $q = \Omega(\Delta^{(b-1)/2}/\varepsilon)$.

Again focusing on the regime where ε is constant, we now obtain $\Omega(\Delta)$ as the lower bound for regimes where $\Delta = O(n^{1/3-c})$ for a constant $c > 0$ or also $\tilde{\Omega}(\Delta)$ when $\Delta = \Theta(n^{1/3})$. Hence, given the algorithm of Theorem 1, up to $\Delta = \Theta(n^{1/3})$ there is essentially *no advantage* for two-sided error algorithms compared to one-sided error ones. For larger values of Δ , the lower bound is $\Omega(\sqrt{n/\Delta})$ and again deteriorates as Δ increases.

Since we are dealing with two-sided error algorithms, we apply Yao’s minimax principle, and we now generate instances according to two different distributions D_Y and D_N where D_Y follows 1-BP whereas D_N generates instances that are (with high probability) far from doing so. The point is that we can show that it is hard to distinguish between D_Y and D_N without making a considerable number of queries. The distributions are such that, in both cases, we pick a set S of $\Theta(\varepsilon n/\Delta)$ vertices in the first time step uniformly at random. Then we color S and $N(S)$ black in D_Y (and leave the remaining nodes white) while in D_N we color only a (constant) fraction of $N(v)$ for $v \in S$ black. (We must also offset the fact that nodes in the second step in D_N are colored black with less probability by using a larger S when generating D_N instances.) By the expansion guarantees, this then gives us ε -farness of the instances in D_N . Observe that in this setting it is meaningless to query nodes in the first step since only a small fraction of them can ever be black; hence we need only deal with a set Q of nodes that are queried in the second step. The indistinguishability of D_Y and D_N follows from using the expansion guarantee from nodes in the second step to those in the first one. The argument is that, unless the set Q of queried nodes is large, almost all neighbors of Q are in fact *unique neighbors* and, moreover, it is impossible to distinguish D_Y from D_N if the set S only intersects the unique neighbors of Q . (That is, one can only distinguish D_Y and D_N if one queries *two* distinct neighbors $u, u' \in N(v)$ of some $v \in S$; due to the expansion guarantees, this requires a large number of queries.)



■ **Figure 1** Summary of results for the case $T = 2$ and constant ε , ignoring logarithmic factors.

In light of these lower bounds, looking back at the algorithm of Theorem 1 we realize that its single weakness is that it does not perform well when Δ is large. Unfortunately our lower bounds do not say as much in that case, and thus a wide gap is left between lower and upper bounds in that regime. Nevertheless, we can narrow this gap by using a more complex strategy – if we are prepared to let go of non-adaptiveness and time-conformability (though we can still obtain a one-sided error algorithm). As previously discussed, this is not such a large limitation when taking possible applications into account (as when $T = 2$ it is plausible to, e.g., require nodes to cache their previous state) and indeed it is offset by the significant reduction in the query complexity.

► **Theorem 4.** *Let $T = 2$ and let $\varepsilon > 0$ be given. There is an adaptive, one-sided error algorithm for testing whether $\text{ENV} \in 1\text{-BP}$ or is ε -far from 1-BP with query complexity $O(\sqrt{n} \log^{3/2}(n)/\varepsilon)$.*

The algorithm achieving this is much more complex than that of Theorem 1. Indeed, it must decide whether $\text{ENV} \in 1\text{-BP}$ or not *without being able to query the entire neighborhood of any node*. To achieve this, we use a “filtering” process in which we first try to infer the color (assuming $\text{ENV} \in 1\text{-BP}$) of as many nodes as we can (in either step) by querying some of their neighbors indirectly. Since we are certain of which color these nodes must have, we can verify these separately using a small number of random queries. Making careful observations, we then realize that we can simply ignore these nodes afterwards and thus reduce the degree of most of the remaining nodes to $\tilde{O}(\sqrt{n})$. This allows us to essentially fall back to a strategy as in the algorithm of Theorem 1, though a particular corner case requires special attention.

The results for $T = 2$ and the regime where ε is constant are summarized in Figure 1.

1.2.2 Case of General T

Let us now discuss the case $T > 2$. Here we obtain a couple of upper bounds that show that the problem admits testing algorithms with sublinear query complexity, at least in a few regimes of interest. We present two algorithms that complement each other.

A quick observation shows the problem becomes essentially trivial when $T \geq 2 \text{diam}(G)/\varepsilon$. (In a nutshell, this is because otherwise $\text{ENV} \in 1\text{-BP}$ reaches a fixed point well before T , and thus most configurations of ENV must all be this one fixed point.) Hence for this discussion it should be kept in mind that the problem is only interesting when $\text{diam}(G)$ is non-trivial and $T = o(\text{diam}(G)/\varepsilon)$. Furthermore, recall that, since ENV has nT entries, the benchmark for a non-trivial testing algorithm is not $o(n)$ but $o(nT)$.

The first algorithm we present is a direct generalization of the one from Theorem 1.

► **Theorem 5.** *Let $\varepsilon > 0$ and $T > 2$. There is a non-adaptive, one-sided error algorithm that performs $O(\Delta^{T-1}/\varepsilon T)$ queries and decides if $\text{ENV} \in 1\text{-BP}$ or $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$.*

The algorithm is only useful in settings where, say, $T = O(\log_{\Delta} n)$. Nevertheless, it is relatively simple to obtain and outperforms our more complex algorithm in certain regimes.

► **Theorem 6.** *Let $\varepsilon > 0$ and $T \geq 4/\varepsilon$. Then there is a non-adaptive, one-sided error algorithm with query complexity $O(|E| \log(n)/\varepsilon T)$ that decides whether $\text{ENV} \in 1\text{-BP}$ or $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$. In addition, if G excludes a fixed minor H (which includes the case where G is planar or, more generally, G has bounded genus), then $O(|E|/\varepsilon T)$ queries suffice.*

To better judge what this algorithm achieves, let us suppose that the underlying graph is Δ -regular, in which case $|E| = n\Delta$. Then this gives a non-trivial testing algorithm whenever $T = \omega(\sqrt{(\Delta/\varepsilon) \log n})$ (or $T = \omega(\sqrt{\Delta/\varepsilon})$ if we also assume G is planar). Hence, together with Theorem 5, we obtain non-trivial testing algorithms in the regime where $\Delta = o(\log n)$ (or even $\Delta = o(\log^2 n)$ in planar graphs) and for all values of T .

The algorithm of Theorem 6 combines some ideas from the work of Nakar and Ron [18] with *graph decompositions*. A graph decomposition is a set C of edges which cuts the graph into components pairwise disjoint components V_1, \dots, V_r of small diameter. In our case the appropriate choice of diameter will be $d = O(\varepsilon T)$. The basic approach is to query the endpoints of C after d steps have elapsed and then use this view to predict the colors of every node in the graph in the subsequent steps. As we show, the view actually suffices to predict all but at most an $O(\varepsilon)$ fraction of ENV (and hence we need only query the predicted values using $O(1/\varepsilon)$ independent queries to check if ENV is following 1-BP or not). We refer to Appendix A.2 for a more in-depth description of the strategy and the ideas involved.

1.3 Open Problems

Since this work is but a first step in an unexplored direction, several questions remain open:

- *The case $T = 2$ and large Δ .* The algorithm of Theorem 1 is essentially optimal up to $\Delta = O(\sqrt{n})$ (if we consider only one-sided error algorithms), but for larger values of Δ the best we have is the $\tilde{O}(\sqrt{n})$ -query algorithm of Theorem 4. Can we reduce this, say, to $\tilde{O}(\Delta^{b-1}/\varepsilon)$ for $\varepsilon = \Omega(\Delta^b/n)$ so as to match the lower bound of Theorem 2? Is it really necessary to give up time-conformity in order to do better than $O(\Delta/\varepsilon)$ in this setting? In addition, improving our lower bounds in the case of (both one- and two-sided error) adaptive algorithms seems well within reach.
- *The case $T > 2$.* Our results show that, in this case, we can get non-trivial algorithms for graphs of small degree (e.g., $\Delta = o(\log n)$). Due to the difficulties in the case $T = 2$, larger values of Δ pose additional challenges. In this sense a first step in this direction would be to port the lower bounds from the $T = 2$ case. Nevertheless, it is not immediately clear how to do so since ε -farness there is even harder to achieve given the cascading effects that might occur over multiple time steps (see in particular Lemma 17).
- *Testing other rules.* Finally, from a broader perspective it would also be meaningful to consider other rules than 1-BP. Of course, by inverting the roles of 0 and 1, all of our results also hold for AND rule (i.e., a node becomes a 1 if and only if all its neighbors are 1; otherwise it becomes a 0). Some very natural rules to consider next are, for instance, τ -BP or the majority rule. There has been extensive study of these rules in other contexts [11, 20, 13, 16, 21, 2, 12, 8, 14], and so there is solid ground to build on there.

1.4 Paper Overview

The rest of the paper is structured as follows: In Section 2 we introduce basic notation, review some standard graph-theoretic results, and formally specify the model and problem we study. For the case $T = 2$, in Section 3 we address the two algorithms (Theorems 1 and 4); the two lower bounds (Theorems 2 and 3) are covered in the full version of the paper [17]. Finally in Appendix A we discuss the two algorithms for the case $T > 2$ (Theorems 5 and 6).

2 Preliminaries

The set of non-negative integers is denoted by \mathbb{N}_0 and that of strictly positive integers by \mathbb{N}_+ . For $n \in \mathbb{N}_+$, we write $[n] = \{i \in \mathbb{N}_+ \mid i \leq n\}$ for the set of the first n positive integers. Without ambiguity, for a statement S , we write $[S]$ for the indicator variable of S (i.e., $[S] = 1$ if S holds; otherwise, $[S] = 0$).

An event is said to occur with high probability if it occurs with probability $1 - o(1)$. For a set X , we write U_X to denote a random variable that takes on values from X following the uniform distribution on X . We assume the reader is familiar with basic notions of discrete probability theory (e.g., Markov's inequality and the union bound). We will use the following version of the Chernoff bound (see, e.g., [9, 19]):

► **Theorem 7 (Chernoff bound).** *Let $n \in \mathbb{N}_+$ and $\varepsilon > 0$, and let X_1, \dots, X_n be independent and identically distributed random variables taking values in the interval $[0, 1]$. Then, for $X = (\sum_{i=1}^n X_i)/n$ and $\mu = \mathbb{E}[X]$, $\Pr[|X - \mu| > \varepsilon] < 2e^{-n\varepsilon^2/3\mu}$.*

2.1 Graph Theory

We consider only undirected graphs. Except when explicitly written otherwise, we always write just “graph” for a simple graph, though self-loops are allowed.

Let $G = (V, E)$ be a graph. For $S \subseteq V$, $G[S]$ denotes the subgraph of G induced by S . For two nodes $u, v \in V$, $\text{dist}_G(u, v)$ is the length of the shortest path between u and v ; we drop the subscript if G is clear from the context. The *diameter* $\text{diam}(G)$ of G is the maximum length among all shortest paths between any pair of vertices $u, v \in V$, that is, $\text{diam}(G) = \max_{u, v \in V} \text{dist}(u, v)$. This notion extends to any $V' \subseteq V$ by considering only pairs of vertices in V' , that is, $\text{diam}(V') = \max_{u, v \in V'} \text{dist}(u, v)$. We write $\delta(G)$ for the minimum degree of G and $\Delta(G)$ for the maximum one. If G is clear from the context, we simply write δ and Δ , respectively. If $\delta = \Delta$, then G is Δ -regular.

For a node $v \in V$, $N(v) = \{u \in V \mid uv \in E\}$ denotes the set of *neighbors* of v . Generalizing this notation, for a set $S \subseteq V$ we write $N(S)$ for the union $\bigcup_{v \in S} N(v)$. A vertex $u \in V$ is said to be a *unique neighbor* of S if there is a *unique* $s \in S$ such that $us \in E$. When S is clear from the context, we also refer to a unique neighbor of $v \in S$ as a node $u \in V$ for which $u \in N(v')$ if and only if $v' \notin S$ or $v' = v$.

A graph $G = (V, E)$ is *bipartite* if $V = L \cup R$ for disjoint sets L and R and any edge has exactly one endpoint in L and one in R . In this context, we refer to the nodes of L as *left-* and to those of R as *right-vertices*. Additionally, the graph is *balanced* if $|L| = |R|$.

The following is a spin-off of a well-known result on the size of the dominating set of a graph (see, e.g., [1]):

► **Lemma 8 (Cover from minimum degree).** *Let $G = (V, E)$ be a bipartite graph where each right-vertex has degree at least δ . Then there is a set D of $n \log(n)/\delta$ left-vertices such that every right-vertex has a neighbor in D .*

Proof. We use the probabilistic method. Fix a right-vertex $v \in V$. If we pick a set D of $m = n \log(n)/\delta$ left-vertices uniformly at random, then the probability that $N(v) \cap D$ is empty is at most $(1 - \delta/n)^m < e^{-\log n} < 1/n$. Hence, by the union bound, there is a non-zero probability that D is such that $N(v) \cap D$ is non-empty for every right-vertex v . ◀

2.2 Model and Problem Definition

We use the standard query model of property testing [5]. The testing algorithm has unlimited computational power and access to a source of infinitely many random bits that are fully independent from one another. In addition, the model has full knowledge of the underlying topology of the network, which is presented as a graph $G = (V, E)$ with $|V| = n$ nodes. We assume there are no singleton nodes (i.e., every node is such that there is an edge incident to it). The topology remains fixed during the evolution of the network, whose nodes take on different states over a set of discrete time steps. As in the previous works [18, 10], the formal object we are testing is an *environment* $\text{ENV}: V \times [T] \rightarrow Z$ where $T \geq 2$ and Z is the set of states that each node may assume.

The goal is to detect whether ENV is following a certain *local rule* ρ , which is defined as a function that maps every multiset μ over Z to $\rho(\mu) \in Z$. The environment ENV is said to *follow* ρ if, for every time step $t \leq T$ and every node $v \in V$, we have that $\text{ENV}(v, t+1) = \rho(\text{ENV}(N(v), t))$ (where $\text{ENV}(N(v), t)$ here is seen as a multiset, that is, counting multiplicities of the occurrence of each element of Z). Blurring the distinction between ρ and the set of environments that follow it, we write $\text{ENV} \in \rho$ if ENV follows ρ .

The *distance* between two environments $\text{ENV}, \text{ENV}': V \times [T] \rightarrow Z$ is the (normalized) number of pairs on which ENV and ENV' differ:

$$\text{dist}(\text{ENV}, \text{ENV}') = \frac{1}{nT} \sum_{(v,t) \in V \times [T]} [\text{ENV}(v,t) \neq \text{ENV}'(v,t)].$$

For a set of environments X (all over the same domain $V \times [t]$), we write $\text{dist}(\text{ENV}, X) = \min_{\text{ENV}' \in X} \text{dist}(\text{ENV}, \text{ENV}')$ for the minimum distance between ENV and X . Being a bit sloppy, we write $\text{dist}(\text{ENV}, \rho)$ for the minimum distance from ENV to the set of environments ENV' for which $\text{ENV}' \in \rho$. For $\varepsilon \geq 0$, ENV is ε -far from ρ if $\text{dist}(\text{ENV}, \rho) \geq \varepsilon$; otherwise it is ε -near ρ .

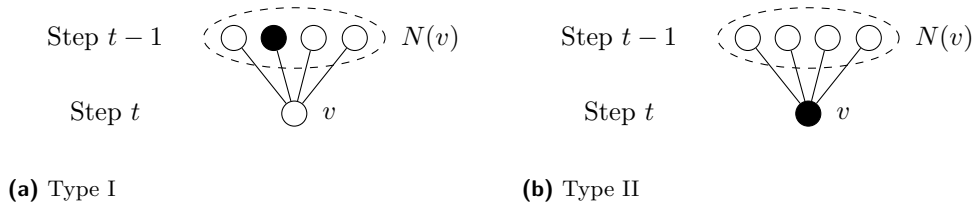
In this work, we focus on $Z = \{0, 1\}$ and on testing the 1-BP rule of bootstrap percolation. The rule is defined by $\rho(\mu) = [1 \in \mu]$ (i.e., $\rho(\mu) = 1$ if $1 \in \mu$ and $\rho(\mu) = 0$ otherwise). Seeing states as colors, we identify state 1 with the color *black* and state 0 with *white*.²

For $t \geq 2$, a pair (v, t) is a *successor* of $(u, t-1)$ if there is an edge between v and u ; at the same time, $(u, t-1)$ is a *predecessor* of (v, t) . If the respective time steps t and $t-1$ are clear from the context, we might also drop any mention of them and simply say that v (as a node) is a successor of u . This is particularly convenient when analyzing the case $T = 2$.

Testing algorithms. Fix $\varepsilon > 0$. A *testing algorithm* A for 1-BP accesses $\text{ENV}: V \times [T] \rightarrow Z$ by means of *queries*, which are pairs $(v, t) \in V \times [T]$. Upon querying the pair (v, t) , A receives $\text{ENV}(v, t)$ as answer. If the queries are performed in an order where, for every t and t' with $t' > t$, A never makes a (\cdot, t) query after it has queried (\cdot, t') , then A is said to be *time-conforming*. As usual in property testing, our interest lies in the *query complexity* of A , that is, the maximum number of queries that A makes, regardless of its randomness.

² Being pedantic, the 1-BP rule in the context of bootstrap percolation is such that a black node always remains black. This behavior can be enforced in the model we describe by adding self-loops to all nodes.

112:10 Testing Spreading Behavior in Networks with Arbitrary Topologies



■ **Figure 2** Violations can be of two different types. Here we see a node v and its state in time step t (as a color) as well as its neighbors $N(v)$ and their respective states in step $t - 1$.

The algorithm A is a *one-sided error tester* for $\text{ENV} \in 1\text{-BP}$ if the following holds, where the probabilities are taken over the randomness of A :

- If $\text{ENV} \in 1\text{-BP}$, then always $A(\text{ENV}) = 1$.
- If ENV is ε -far from 1-BP , then $\Pr[A(\text{ENV}) = 1] < 1/2$.

In contrast, A is a *two-sided error tester* if it may also err on $\text{ENV} \in 1\text{-BP}$:

- If $\text{ENV} \in 1\text{-BP}$, then $\Pr[A(\text{ENV}) = 1] \geq 2/3$.
- If ENV is ε -far from 1-BP , then $\Pr[A(\text{ENV}) = 1] < 1/3$.

Violations. Observe that our notion of distance is *not* the same as counting the number of failures of ENV in following 1-BP . There are two kinds of failures that may occur:

► **Definition 9 (Violations).** A pair $(v, t) \in V \times [T]$ is violating if $t \geq 2$ and one of the following conditions hold:

- (I) $\text{ENV}(v, t) = 0$ and $\exists u \in N(v) : \text{ENV}(u, t - 1) = 1$
- (II) $\text{ENV}(v, t) = 1$ and $\forall u \in N(v) : \text{ENV}(u, t - 1) = 0$

We refer to these violations as violations of type I and II, respectively. We write $\text{viol}(\text{ENV})$ for the set of violating pairs in ENV .

Although a larger distance to 1-BP implies a greater number of violations, there is not an exact correspondence between the two. For example, it might be the case that ENV exhibits a great number of violations, but correcting them requires recoloring only a few nodes. We will prove upper and lower bounds between the distance and the number of violations further below (Lemmas 10 and 17).

3 Upper Bounds for the Case $T = 2$

In this section we present our two algorithms for the case where $T = 2$. The first of these (Section 3.1) is quite simple and has query complexity $O(\Delta/\varepsilon)$, which turns out to be optimal for the regimes where $\Delta = o(\sqrt{n})$. The second one (Section 3.2) is much more intricate and gives query complexity $\tilde{O}(\sqrt{n}/\varepsilon)$, which makes it more suitable for the regimes where $\Delta = \omega(\sqrt{n})$. Although both are one-sided error algorithms, the first algorithm is non-adaptive and thus time-conforming whereas the second has neither of these properties (i.e., it is adaptive and also does not respect time-conformity).

3.1 An Upper Bound that Scales with the Maximum Degree

In this section, we prove:

► **Theorem 1.** Let $T = 2$ and $\varepsilon > 0$. There is a non-adaptive, one-sided error algorithm with query complexity $O(\Delta/\varepsilon)$ that decides whether $\text{ENV} \in 1\text{-BP}$ or $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$.

The claim is that Algorithm 1 satisfies the requirements of Theorem 1. As mentioned above, the strategy followed by Algorithm 1 is quite simple: It chooses a certain subset of nodes U uniformly at random and then queries the states of $u \in U$ and all of $N(u)$ in both time steps. The algorithm then rejects if and only if a violation of either type is detected.

■ **Algorithm 1** Algorithm for the case $T = 2$ with query complexity $O(\Delta/\varepsilon)$.

```

1 Pick  $U \subseteq V$  uniformly at random where  $|U| = \lceil 2/\varepsilon \rceil$ ;
2 Query  $\text{ENV}(v, 1)$  and  $\text{ENV}(u, 2)$  for every  $u \in U$  and  $v \in N(u)$  in a time-conforming
   manner;
3 for  $u \in U$  do
4   | if  $\text{ENV}(u, 2) = 0$  and  $\exists v \in N(u) : \text{ENV}(v, 1) = 1$  then reject;
5   | if  $\text{ENV}(u, 2) = 1$  and  $\forall v \in N(u) : \text{ENV}(v, 1) = 0$  then reject;
6 end
7 accept;
```

At the core of the correctness of Algorithm 1 is the relation between the number of violations and the distance of ENV to 1-BP. With a bit of care, we can relate the two quantities as shown next. (Actually for the correctness of Algorithm 1 we only need one of the two bounds below; the other one comes as a “bonus”.)

► **Lemma 10.** *Let $T = 2$. Then*

$$\frac{|\text{viol}(\text{ENV})|}{2\Delta n} \leq \text{dist}(\text{ENV}, 1\text{-BP}) \leq \frac{|\text{viol}(\text{ENV})|}{2n}.$$

Proof. Every violating pair (u, t) can be corrected by flipping the value of $\text{ENV}(u, t)$, which does not create a new violating pair since $t = T = 2$. In addition, if ENV does not have any violating pair, then $\text{ENV} \in 1\text{-BP}$. This implies $\text{dist}(\text{ENV}, 1\text{-BP}) \leq |\text{viol}(\text{ENV})|/2n$. On the other hand, flipping the color of a node can only correct at most Δ violating pairs. Hence we also have $\text{dist}(\text{ENV}, 1\text{-BP}) \geq |\text{viol}(\text{ENV})|/2\Delta n$. ◀

The lemma directly implies that, if $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$, then $|\text{viol}(\text{ENV})| \geq 2\varepsilon n$. Hence the probability that Algorithm 1 errs in this case is

$$\Pr[(U, 2) \cap \text{viol}(\text{ENV}) = \emptyset] \leq (1 - 2\varepsilon)^{|U|} < \frac{1}{e} < \frac{1}{2}.$$

Since Algorithm 1 only rejects when a violation of either type is detected, Algorithm 1 always accepts if $\text{ENV} \in 1\text{-BP}$. The query complexity and other properties of Algorithm 1 are clear, and hence Theorem 1 follows.

3.2 An Upper Bound Independent of the Maximum Degree

Next we show our second algorithm, which is much more complex than Algorithm 1. Since Algorithm 1 is already optimal for $\Delta = O(\sqrt{n})$, we focus on the regime where $\Delta = \Omega(\sqrt{n})$ and present an algorithm with query complexity that is independent of Δ . The algorithm requires adaptiveness and unfortunately is no longer time-conforming; obtaining a time-conforming or even non-adaptive algorithm with the same query complexity for these large values of Δ (or proving none exists) remains an interesting open question.

► **Theorem 4.** *Let $T = 2$ and let $\varepsilon > 0$ be given. There is an adaptive, one-sided error algorithm for testing whether $\text{ENV} \in 1\text{-BP}$ or is ε -far from 1-BP with query complexity $O(\sqrt{n} \log^{3/2}(n)/\varepsilon)$.*

112:12 Testing Spreading Behavior in Networks with Arbitrary Topologies

We claim Algorithm 2 satisfies the requirements of the theorem. Next we give a brief description of the strategy followed by Algorithm 2.

■ **Algorithm 2** Algorithm for the case $T = 2$ with query complexity $\tilde{O}(\sqrt{n}/\varepsilon)$.

-
- 1 Select $Q_1, Q'_1, Q_2, Q'_2 \subseteq V$ with $|Q_i| = |Q'_i| = (24/\varepsilon)\sqrt{n} \log^{3/2} n$ uniformly at random;
 - 2 Query $\text{ENV}(Q_1, 1)$ and $\text{ENV}(Q'_2, 1)$;
 - 3 $B_2 \leftarrow \{v \in V \mid \exists u \in N(v) \cap Q_1 : \text{ENV}(u, 1) = 1\}$;
 - 4 Query $\text{ENV}(Q'_1, 2)$ and $\text{ENV}(Q_2, 2)$;
 - 5 $W_1 \leftarrow \{v \in V \mid \exists u \in N(v) \cap Q_2 : \text{ENV}(u, 2) = 0\}$;
 - 6 **if** $\exists u \in Q'_1 \cap B_2 : \text{ENV}(u, 2) = 0$ **or** $\exists u \in Q'_2 \cap W_1 : \text{ENV}(u, 1) = 1$ **then reject**;
 - 7 $F \leftarrow \{v \in V \mid |N(v) \setminus W_1| \leq 4\sqrt{n} \log n\}$;
 - 8 Select $Q_3 \subseteq F$ with $|Q_3| = (4/\varepsilon) \log n$ uniformly at random;
 - 9 Query $\text{ENV}(v, 2)$ and $\text{ENV}(N(v) \setminus W_1, 1)$ for every $v \in Q_3$;
 - 10 **if** $\exists v \in Q_3 : \text{ENV}(v, 2) = 0 \wedge \exists u \in N(v) \setminus W_1 : \text{ENV}(u, 1) = 1$ **or**
 $\exists v \in Q_3 : \text{ENV}(v, 2) = 1 \wedge \nexists u \in N(v) \setminus W_1 : \text{ENV}(u, 1) = 1$ **then reject**;
 - 11 **accept**;
-

Approach. The operation of Algorithm 2 can be divided into two parts. The first one is up to line 2. Here we query nodes from the first and second time steps at random (Q_1 and Q_2) and try to ascertain the color of as many nodes as possible using these queries. More specifically, if a node v has a neighbor $u \in N(v)$ which is black in the first step, then we know v must be black in the second step. We gather these nodes in the set B_2 . A similar observation holds for the nodes in the set W_1 , which must be white since they have a neighbor in the second step that is white. At the same time we query another set of nodes from the first and second step uniformly at random (Q'_1 and Q'_2) to verify that all but a very small fraction of nodes in W_1 (resp., B_2) are indeed white (resp., black).

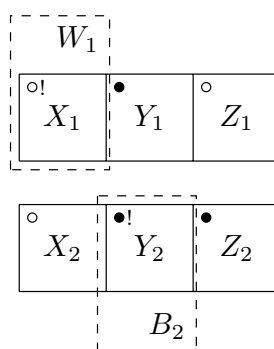
The second part of the algorithm starts after line 2. Here we will ignore nodes in W_1 (since we already know they are white) and “filter” nodes that have not too large degree to nodes not in W_1 . These nodes are added to the set F . Intuitively we can then test these nodes in the same fashion as Algorithm 1: We select a few nodes $v \in F$ uniformly at random (Q_3) and then query the entire neighborhood of these nodes in the first step, so $\text{ENV}(u, 1)$ for $u \in N(v) \setminus W_1$, as well as $\text{ENV}(v, 2)$. If any violations are detected here, then we can safely reject. What then remains are only nodes with high degree; as we argue in the analysis below, any set of nodes not in F that are black (which might occur when $\text{ENV} \notin 1\text{-BP}$) and which have no white predecessor can actually be covered by recoloring only a small set of nodes (and hence ENV must be close to 1-BP).

Analysis. The query complexity of Algorithm 2 is clear, so we focus on the analysis on its correctness. First we show that Algorithm 2 is indeed a one-sided error algorithm; that is:

▷ **Claim 11.** If $\text{ENV} \in 1\text{-BP}$, then Algorithm 2 always accepts.

Intuitively this is the case because we are only trying to detect violations (and accept unconditionally if we do not manage to find any).

Proof. Since $\text{ENV} \in 1\text{-BP}$, we have $\text{ENV}(W_1, 1) = 0$ and $\text{ENV}(B_2, 2) = 1$. As a result, Algorithm 2 never rejects in line 2. Consider the two possibilities for Algorithm 2 to reject in line 2. The first is that there is a node $v \in Q_3$ with $\text{ENV}(v, 2) = 0$ and some $u \in N(v)$



■ **Figure 3** Relation between the sets used in the analysis of Algorithm 2. The sets form a partition of the nodes in the first and second time steps. The small circles indicate the color of the nodes in each set or, in the case of X_1 or Y_2 , that the algorithm rejects unless (almost all) nodes in the set have the respective color (denoted with an exclamation mark).

so that $\text{ENV}(u, 1) = 1$, which contradicts $\text{ENV} \in 1\text{-BP}$. The second is that $\text{ENV}(v, 2) = 1$ and $\text{ENV}(u, 1) = 0$ for every $u \in N(v) \setminus W_1$; however, since $\text{ENV}(W_1, 1) = 0$, this means $\text{ENV}(u, 1) = 0$ for every $u \in N(v) \cap W_1$ as well and then $\text{ENV}(u, 1) = 0$ for every $u \in N(v)$, thus also contradicting $\text{ENV} \in 1\text{-BP}$. \triangleleft

Now we turn to proving that Algorithm 2 does not have false positives. More specifically we show that Algorithm 2 can only accept with constant probability if $\text{dist}(\text{ENV}, 1\text{-BP}) < \varepsilon$ is the case (and so, conversely, Algorithm 2 rejects with high probability if $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$).

For $v \in V$ and $t \in \{1, 2\}$, we write $\text{bn}_t(v)$ and $\text{wn}_t(v)$ for the number of black and white neighbors, respectively, of v in step t ; formally,

$$\text{bn}_t(v) = |\{u \in N(v) \mid \text{ENV}(u, t) = 1\}|, \quad \text{wn}_t(v) = |\{u \in N(v) \mid \text{ENV}(u, t) = 0\}|.$$

Let $\theta = (\varepsilon/4)\sqrt{n/\log n}$ and consider the following sets:

$$\begin{aligned} X_1 &= \{v \in V \mid \text{wn}_2(v) \geq \theta\}, & X_2 &= \{v \in V \mid \text{bn}_1(v) < \theta \wedge \text{ENV}(v, 2) = 0\}, \\ Y_1 &= \{v \in V \mid \text{wn}_2(v) < \theta \wedge \text{ENV}(v, 1) = 1\}, & Y_2 &= \{v \in V \mid \text{bn}_1(v) \geq \theta\}, \\ Z_1 &= \{v \in V \mid \text{wn}_2(v) < \theta \wedge \text{ENV}(v, 1) = 0\}, & Z_2 &= \{v \in V \mid \text{bn}_1(v) < \theta \wedge \text{ENV}(v, 2) = 1\}. \end{aligned}$$

The sets X_1 and Y_2 contain the nodes for which we can detect that they must be white and black, respectively, by using the query sets Q_1 and Q_2 .

▷ **Claim 12.** With high probability over the choices made by Algorithm 2, $X_1 \subseteq W_1$ and $Y_2 \subseteq B_2$.

Proof. Fix $v \in X_1$. By the Chernoff bound, the probability that no $u \in N(v)$ with $\text{ENV}(u, 2) = 0$ lands in Q_2 is at most $2e^{-2 \log n} < 2/n^2$. Hence, by the union bound, the probability that $X_1 \setminus W_1$ is non-empty is $O(1/n)$. The same applies to Y_2 and B_2 . \triangleleft

Next we observe that the queries from Q'_1 and Q'_2 significantly “reduce” the number of black or white nodes in X_1 or Y_2 , respectively; that is, if there is a significant number of such nodes in these sets, then Algorithm 2 will detect them anyway and reject (and thus we can focus the analysis on instances where this is not the case).

▷ **Claim 13.** If ENV is such that there are $\varepsilon\sqrt{n}$ nodes $v \in X_1$ with $\text{ENV}(v, 1) = 1$ or $\varepsilon\sqrt{n}$ nodes $v \in Y_2$ with $\text{ENV}(v, 2) = 0$, then Algorithm 2 rejects ENV with high probability.

112:14 Testing Spreading Behavior in Networks with Arbitrary Topologies

Proof. Let $S \subseteq V$ be a subset of $|S| \geq \varepsilon\sqrt{n}$ vertices. Then the probability that $S \cap Q'_i$ is empty is at most

$$\left(1 - \frac{\varepsilon}{\sqrt{n}}\right)^{(24/\varepsilon)\sqrt{n}\log^{3/2}n} < e^{-24\log^{3/2}n} = o\left(\frac{1}{n}\right).$$

Using Claim 12, we have $X_1 \subseteq W_1$ and $Y_2 \subseteq B_2$ with high probability. In this case Algorithm 2 rejects if any node $v \in X_1 \subseteq W_1$ with $\text{ENV}(v, 1) = 1$ lands in Q'_2 or any $v \in Y_2 \subseteq B_2$ with $\text{ENV}(v, 2) = 0$ lands in Q'_1 . Therefore Algorithm 2 rejects with high probability if there are at least $\varepsilon\sqrt{n}$ nodes of either type. \triangleleft

Hence we may now safely assume that all but at most $O(\varepsilon\sqrt{n})$ nodes in X_1 are white in the first time step and that all but at most $O(\varepsilon\sqrt{n})$ nodes in Y_2 are black in the second one. The next observation is that nodes in X_2 are highly connected to X_1 . This justifies filtering nodes based on their connections to $W_1 \supseteq X_1$.

\triangleright **Claim 14.** On average, a node from X_2 has at most $\theta n/|X_2|$ neighbors not in X_1 .

Proof. Every node in Y_1 or Z_1 has at most θ white neighbors by definition, so at most this many neighbors in X_2 . Hence there are at most θn edges in total between X_2 and nodes not in X_1 . \triangleleft

Finally we show that, if Algorithm 2 accepts ENV with at least constant probability, then we can correct all violations of either type with at most $\varepsilon n/2$ modifications in total for each type. In both cases we must be careful so that these modifications do not create new violations of their own.

\triangleright **Claim 15.** If Algorithm 2 accepts ENV with at least constant probability, then there are at most $\varepsilon n/2$ many type I violations in ENV. These violations can be corrected (without creating any new ones) by recoloring $\text{ENV}(v, 2)$ black for every violation $(v, 2)$.

Proof. Let R be the set of nodes corresponding to type I violations, that is, $R = \{v \in V \mid \text{ENV}(v, 2) = 0 \wedge \exists u \in N(v) : \text{ENV}(u, 1) = 1\}$. We prove the claim by proving the contrapositive; that is, if $|R| \geq \varepsilon n/2$, then Algorithm 2 rejects ENV with high probability.

The first observation is that we have $|R \setminus X_2| = o(\varepsilon n)$ (with high probability) due to Claim 13 and then, by the assumption on R , $|X_2| \geq (1 - o(1))\varepsilon n/2$. Hence we focus our analysis on $R \cap X_2$. By Claim 14, on average a node from X_2 has at most $2\theta/\varepsilon = (1/2)\sqrt{n/\log n}$ many neighbors that are outside X_1 . By Markov's inequality, this gives us that there are at most $O(n/\log n)$ many nodes $v \in X_2$ for which $|N(v) \setminus X_1| > 4\sqrt{n \log n}$. Using Claim 12, we have $X_1 \subseteq W_1$ and so altogether we have $|R \cap F| \geq \varepsilon n/4$ (with high probability). In this case the probability that $R \cap Q_3$ is empty is at most $(1 - \varepsilon/4)^{4 \log(n)/\varepsilon} < e^{-\log(n)} = O(1/n)$, and so Algorithm 2 rejects with high probability. \triangleleft

\triangleright **Claim 16.** If Algorithm 2 accepts ENV with at least constant probability, then all type II violations in ENV can be corrected by recoloring at most $\varepsilon n/2$ nodes. In particular, this recoloring is such that we color $\text{ENV}(v, 1)$ and $\text{ENV}(N(v), 2)$ black for a certain subset of nodes v (and hence does not create any new violations).

Proof. Similar to the proof of Claim 15, let $R = \{v \in V \mid \text{ENV}(v, 2) = 1 \wedge \forall u \in N(v) : \text{ENV}(u, 1) = 0\}$ be the set of type II violations. We show that, if Algorithm 2 accepts with at least constant probability, then we can correct R by recoloring at most $\varepsilon n/2$ nodes black. (Note this does not necessarily mean that $|R| < \varepsilon n/2$ as in the proof of Claim 15. Instead what we prove is an upper bound on the number of recolorings needed to correct R .)

By Claim 13, $|R \setminus Z_2| = o(\varepsilon n)$ and thus $|Z_2| \geq (1 - o(1))\varepsilon n/2$. Arguing as in the proof of Claim 15, if $|R \cap F| \geq \varepsilon n/4$, then Algorithm 2 must reject with high probability. Hence let us focus on the nodes in $R' = R \setminus F$. Consider the bipartite graph where the set of left-vertices is $V \setminus W_1$, that of right-vertices is R' , and the edges are as in G . Then the minimum degree of R' in this graph is $4\sqrt{n \log n}$, which means we can apply Lemma 8 and obtain a cover $D \subseteq V \setminus W_1$ of R' with $|D| = (1/4)\sqrt{n \log n}$ nodes. By Claim 12, $D \cap X_1 = \emptyset$ and hence $\text{wn}_2(v) < \theta$ for every $v \in D$ (with high probability). Therefore we can correct R' by coloring $\text{ENV}(D, 1)$ and $\text{ENV}(N(D), 2)$ all black, which means coloring at most $(\theta/4)\sqrt{n \log n} \leq \varepsilon n/16$ nodes black. Together with $|R \cap F| < \varepsilon n/4$, this means we must color at most $(1/4 + 1/16)\varepsilon n < \varepsilon n/2$ many nodes black in total in order to correct R . \triangleleft

References

- 1 Noga Alon and Joel H. Spencer. *The Probabilistic Method, Third Edition*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 2008.
- 2 József Balogh and Boris G. Pittel. Bootstrap percolation on the random regular graph. *Random Struct. Algorithms*, 30(1-2):257–286, 2007. doi:10.1002/RSA.20158.
- 3 Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–193, 1996. doi:10.1109/SFCS.1996.548477.
- 4 Arnab Bhattacharyya and Yuichi Yoshida. An algebraic characterization of testable boolean CSPs. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 7965, pages 123–134, 2013. doi:10.1007/978-3-642-39206-1_11.
- 5 Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing - Problems and Techniques*. Springer, 2022. doi:10.1007/978-981-16-8622-1.
- 6 Hubie Chen, Matt Valeriote, and Yuichi Yoshida. Constant-query testability of assignments to constraint satisfaction problems. *SIAM J. Comput.*, 48(3):1022–1045, 2019. doi:10.1137/18M121085X.
- 7 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 474–483, 2002. doi:10.1145/509907.509977.
- 8 Silvio Frischknecht, Barbara Keller, and Roger Wattenhofer. Convergence in (social) influence networks. In *Proceedings of the 27th International Symposium on Distributed Computing (DISC)*, pages 433–446, 2013. doi:10.1007/978-3-642-41527-2_30.
- 9 Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.
- 10 Oded Goldreich and Dana Ron. On learning and testing dynamic environments. *J. ACM*, 64(3):21:1–21:90, 2017. doi:10.1145/3088509.
- 11 Paolo De Gregorio, Aonghus Lawlor, and Kenneth A. Dawson. Bootstrap percolation. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 608–626. Springer, 2009. doi:10.1007/978-0-387-30440-3_41.
- 12 Bernd Gärtner and Ahad N. Zehmakan. Majority model on random regular graphs. In *Proceedings of the 13th Latin American Theoretical Informatics Symposium (LATIN)*, pages 572–583, 2018. doi:10.1007/978-3-319-77404-6_42.
- 13 Svante Janson, Tomasz Luczak, Tatyana Turova, and Thomas Vallier. Bootstrap percolation on the random graph $g_{n,p}$. *Ann. Appl. Probab.*, 22(5):1989–2047, 2012. doi:10.1214/11-AAP822.
- 14 Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale. On the voting time of the deterministic majority process. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 55:1–55:15, 2016. doi:10.4230/LIPIcs.MFCS.2016.55.

- 15 Philip N. Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 682–690, 1993. doi:10.1145/167088.167261.
- 16 Diego Maldonado, Pedro Montealegre, Martín Ríos Wilson, and Guillaume Theyssier. Local certification of majority dynamics. In Henning Fernau, Serge Gaspers, and Ralf Klasing, editors, *SOFSEM 2024: Theory and Practice of Computer Science - 49th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2024, Cochem, Germany, February 19-23, 2024, Proceedings*, volume 14519 of *Lecture Notes in Computer Science*, pages 369–382. Springer, 2024. doi:10.1007/978-3-031-52113-3_26.
- 17 Augusto Modanese and Yuichi Yoshida. Testing spreading behavior in networks with arbitrary topologies. *CoRR*, abs/2309.05442, 2023. arXiv:2309.05442.
- 18 Yonatan Nakar and Dana Ron. Testing dynamic environments: Back to basics. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 98:1–98:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.98.
- 19 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.
- 20 Ahad N. Zehmakan. Tight bounds on the minimum size of a dynamic monopoly. In *Proceedings of the 13th International Conference on Language and Automata Theory and Applications (LATA)*, pages 381–393, 2019. doi:10.1007/978-3-030-13435-8_28.
- 21 Ahad N. Zehmakan. Majority opinion diffusion in social networks: An adversarial approach. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pages 5611–5619, 2021. doi:10.1609/aaai.v35i6.16705.

A Upper Bounds for the Case $T > 2$

In this appendix we consider two different strategies for the case where $T > 2$. An immediate observation to make is that the diameter $\text{diam}(G)$ plays a much more significant role in this setting. For instance, the case where $T \geq (1 + 2/\varepsilon) \text{diam}(G)$ is more or less trivial since then after $\text{diam}(G)$ steps every connected component must be either all-black or all-white and the first $\text{diam}(G)$ steps constitute at most an $\varepsilon/2$ fraction of ENV.

A.1 Structure-independent Upper Bound

First we give a generalization of Theorem 1, which is simple to obtain and adequate for settings where Δ and T are not too large. For constant Δ , for instance, it still gives a sublinear query algorithm when $T = O(\log n)$. As the algorithm of Theorem 1, it does not use the graph structure in any way except for determining the neighborhood of each node.

► **Theorem 5.** *Let $\varepsilon > 0$ and $T > 2$. There is a non-adaptive, one-sided error algorithm that performs $O(\Delta^{T-1}/\varepsilon T)$ queries and decides if $\text{ENV} \in 1\text{-BP}$ or $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$.*

We adapt Algorithm 1 to obtain Algorithm 3. The analysis does not carry over automatically since we need to consider what happens if we are correcting violations in a time step $t < T$. Unlike in Lemma 10, this kind of correction may now propagate to time steps after t . In addition, we have to assume $\Delta \geq 2$; however, the case $\Delta = 1$ is trivial since then $\text{diam}(G) = 1$ and we need only follow the strategy described at the beginning of this section.

► **Lemma 17.** *Let $\Delta \geq 2$. Then*

$$\frac{|\text{viol}(\text{ENV})|}{(\Delta + 1)nT} \leq \text{dist}(\text{ENV}, 1\text{-BP}) \leq \frac{\Delta^{T-1} - 1}{(\Delta - 1)nT} |\text{viol}(\text{ENV})|.$$

■ **Algorithm 3** Structure-independent algorithm for the case of general T with query complexity $O(\Delta^{T-1}/\varepsilon)$.

```

1 Pick  $Q \subseteq V \times \{t \mid 2 \leq t \leq T\}$  uniformly at random where  $|Q| = \lceil 2\Delta^{T-2}/\varepsilon T \rceil$ ;
2 Query  $\text{ENV}(v, t-1)$  and  $\text{ENV}(u, t)$  for every  $(u, t) \in Q$  and  $v \in N(u)$  in a
   time-conforming manner;
3 for  $(u, t) \in Q$  do
4   | if  $\text{ENV}(u, t) = 0$  and  $\exists v \in N(u) : \text{ENV}(v, t-1) = 1$  then reject;
5   | if  $\text{ENV}(u, t) = 1$  and  $\forall v \in N(u) : \text{ENV}(v, t-1) = 0$  then reject;
6 end
7 accept;
```

See the full version of the paper [17] for the proof.

Now as before with Theorem 1 we have that $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$ implies

$$\text{viol}(\text{ENV}) \geq \frac{\varepsilon(\Delta-1)nT}{\Delta^{T-1}-1} > \frac{\varepsilon nT}{2\Delta^{T-2}}.$$

Hence the probability that Algorithm 3 errs is

$$\Pr[Q \cap \text{viol}(\text{ENV}) = \emptyset] \leq \left(1 - \frac{\varepsilon T}{2\Delta^{T-2}}\right)^{|Q|} < \frac{1}{e} < \frac{1}{2}.$$

As was the case with Algorithm 1, the query complexity and other properties required in Theorem 5 are clear, and hence Theorem 5 follows.

A.2 Upper Bound Based on Graph Decompositions

The second algorithm is suited for larger values of T and not too dense graphs.

► **Theorem 6.** *Let $\varepsilon > 0$ and $T \geq 4/\varepsilon$. Then there is a non-adaptive, one-sided error algorithm with query complexity $O(|E| \log(n)/\varepsilon T)$ that decides whether $\text{ENV} \in 1\text{-BP}$ or $\text{dist}(\text{ENV}, 1\text{-BP}) \geq \varepsilon$. In addition, if G excludes a fixed minor H (which includes the case where G is planar or, more generally, G has bounded genus), then $O(|E|/\varepsilon T)$ queries suffice.*

The strategy followed by the algorithm relies on graph decompositions. These are partitions induced by sets of edges that cut the graph into components of bounded diameter.

► **Definition 18.** *Let $d \in \mathbb{N}_+$ and $\alpha > 0$. A (d, α) -decomposition of a graph $G = (V, E)$ is a set of edges $C \subseteq E$ with $|C| \leq \alpha|E|$ and such that there is a partition $V = V_1 + \dots + V_r$ satisfying the following:*

1. For $u, v \in V$, $uv \in C$ if and only if there are i and j such that $i \neq j$, $u \in V_i$, and $v \in V_j$.
2. For every i , $\text{diam}(V_i) \leq d$.

The following is a renowned result in graph decompositions:

► **Theorem 19** ([3]). *For any $d \in \mathbb{N}_+$, every graph G admits a $(d, O(\log(n)/d))$ -decomposition.*

This trade-off is optimal for graphs in general. For the special case of graphs excluding a fixed minor (which includes most notably planar graphs or also graphs of bounded genus), we have the following small improvement:

► **Theorem 20** ([15]). *Let H be a fixed graph. For any $d \in \mathbb{N}_+$, every graph G excluding H as a minor admits a $(d, O(1/d))$ -decomposition.*

112:18 Testing Spreading Behavior in Networks with Arbitrary Topologies

■ **Algorithm 4** Algorithm for the case of general T based on network decompositions.

```

1  $t_1 \leftarrow \lfloor \varepsilon T / 4 \rfloor$ ;
2 Compute a  $(t_1, \alpha)$ -decomposition of  $G$  according to Theorem 19 or Theorem 20 and
   obtain a set of edges  $C$  that cuts  $G$  into components  $V_1, \dots, V_r$  as in Definition 18;
3  $B \leftarrow \{v \mid v \text{ is incident to an edge in } C\}$ ;
4 Pick  $Q \subseteq \{(v, t) \mid v \in V_i \text{ and } t \geq t_1\}$  uniformly at random where  $|Q| = \lceil 3/\varepsilon \rceil$ ;
5  $Q' \leftarrow \{v \in V \mid \exists t : (v, t) \in Q\}$ ;
6 Query  $\text{ENV}(B, t_1)$ ,  $\text{ENV}(Q)$ , and  $\text{ENV}(Q', t_1)$  in a time-conforming fashion;
7 if  $\text{ENV}(B, t_1)$  is not feasible then reject;
8 for  $i \in [r]$  do
9    $B_i \leftarrow B \cap V_i$ ;
10   $B'_i \leftarrow \{u \in B_i \mid \text{ENV}(u, t_1) = 1\}$ ;
11 end
12 for  $v \in V$  do
13   for  $i \in [r]$  do
14     if  $B'_i \neq \emptyset$  then
15        $\alpha_i(v) \leftarrow \min_{u \in B'_i \cup (V_i \setminus B_i)} \text{dist}(u, v)$ ;
16        $\beta_i(v) \leftarrow \min_{u \in B'_i} \text{dist}(u, v)$ ;
17     else
18        $\alpha_i(v) \leftarrow \infty$ ;
19        $\beta_i(v) \leftarrow \infty$ ;
20     end
21   end
22    $\alpha(v) \leftarrow \min_i \alpha_i(v)$ ;
23    $\beta(v) \leftarrow \min_i \beta_i(v)$ ;
24 end
25 for  $(v, t) \in Q$  do
26   Let  $i$  be such that  $v \in V_i$ ;
27   if  $\text{ENV}(v, t_1) = 1$  then
28     if  $\text{ENV}(v, t) \neq 1$  then reject;
29   else
30     if  $t_1 \leq t < t_1 + \alpha(v)$  and  $\text{ENV}(v, t) \neq 0$  then reject;
31     if  $t \geq t_1 + \beta(v)$  and  $\text{ENV}(v, t) \neq 1$  then reject;
32   end
33 end
34 accept;
```

The claim is that Algorithm 4 satisfies the requirements of Theorem 6. As mentioned in the introduction, the strategy followed by the algorithm is loosely based on a similar testing routine from the paper by Nakar and Ron [18]. In a nutshell, the idea is to split the environment into more manageable components and then use the properties of the local rule to predict how each component must behave.

Approach. Let us recall the relevant details of the strategy of Nakar and Ron [18]. In their paper, the authors studied local rules resembling the majority rule in the restricted setting where G is a path. Their idea involved splitting the path into intervals that intersect at

periodic control points. The first queries yield the state of these control points at a certain time step t_1 . If there is no initial configuration leading to what we observe at t_1 (i.e., the configuration is not *feasible*), then we can immediately reject. Otherwise we can use the states of the nodes at the control points (plus some additional queries) to fully predict almost the entirety of ENV after t_1 . Hence we only need to test a certain number of times if $\text{ENV}(v, t)$ matches our prediction where $(v, t) \in V \times \{t \in \mathbb{N}_+ \mid t \geq t_1\}$ is chosen uniformly at random.

Our approach is more or less the same, though we need to cater for a couple differences between our setting and theirs. We are not in a path, and so in general we cannot split our graph into intervals of the same size; rather we must work with a graph decomposition, which does give us the adequate control points (the vertices incident to the edges of the cut C , which form the set B in Algorithm 4) but only an upper bound on the diameter of each component (which correspond to the intervals in the setting of Nakar and Ron [18]). Fortunately the 1-BP rule is much simpler than majority or the like, and hence the prediction in each component is easier to make. The relevant observation is that the 1-BP rule converges fast to an (all-black) fixed point in graphs of small diameter. (Indeed, the 1-BP rule converges in at most $\text{diam}(G)$ steps.) More specifically, components that started in an all-zero configuration must stay zero until they enter in contact with a black node; meanwhile a component V_i that had at least one black node in it will converge to an all-black configuration in at most $\text{diam}(V_i) \leq t_1$ steps.

Let us now give a more detailed overview of the steps performed by Algorithm 4. For a set $S \subseteq V$ and $t \in [T]$, we say that $\text{ENV}(S, t)$ is *feasible* if there is $\text{ENV}' \in 1\text{-BP}$ such that $\text{ENV}'(v, t) = \text{ENV}(v, t)$ for every $v \in S$. Theorem 6 first sets t_1 appropriately and determines a graph decomposition of G where the components V_1, \dots, V_r have diameter at most t_1 . We wait for t_1 steps to elapse and then query the states of B , which are the nodes incident to the edge cut C of the graph decomposition, and can immediately reject if what we see is not feasible. At the same time we query a uniformly sampled set Q of pairs corresponding to the states of nodes in time steps after t_1 , whose values we shall use later. We then set $B_i = B \cap V_i$ and B'_i to the nodes that are black in B_i in time step t_1 . With these we can then compute estimates $\alpha_i(v)$ and $\beta_i(v)$ for each node v and each component V_i . These are only intended to be useful if v is white in time step t_1 and are determined as follows:

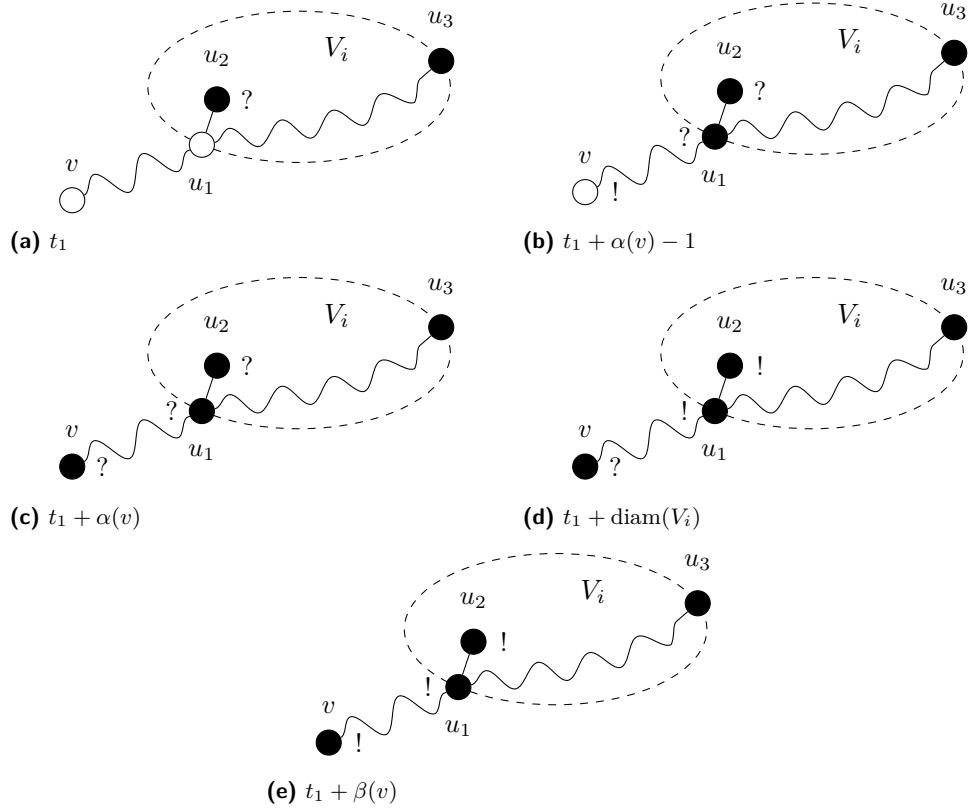
- $\alpha_i(v)$ is a *lower bound* on the number of time steps that elapse after t_1 until v turns from white to black. To compute $\alpha_i(v)$, we consider both nodes in B'_i (whose state in t_1 is known to us) and nodes in the inside of V_i (whose state is unknown and which means we must assume that they are black). If there are no nodes in B'_i , then we know that V_i was all white at the beginning and we just set $\alpha_i(v) = \infty$.
- $\beta_i(v)$ is an *upper bound* on the number of time steps after t_1 until v turns black at the latest. To compute $\beta_i(v)$ we take into account only nodes which we are sure that are black in t_1 , that is, nodes in B'_i . Again, if B'_i is empty, then V_i must have been all white in the first time step; in that case we set $\beta_i(v) = \infty$.

See Figure 4 for an example. Based on these estimates, we can then use the values of Q to make random tests on the state of nodes after t_1 based on what we know from B'_i and $\alpha(v) = \min_i \alpha_i(v)$ and $\beta(v) = \min_i \beta_i(v)$. More precisely, for a pair $(v, t) \in Q$:

- If v was already black in time step t_1 , then it must still be black in time step $t \geq t_1$.
- Otherwise v was white in time step t_1 and we can use our estimates $\alpha(v)$ and $\beta(v)$ to verify the predicted state of v in step t , if possible.

The algorithm accepts by default if ENV passes the tests.

The query complexity of Theorem 6 is evident. We refer for the full version of the paper [17] for the proof of its correctness.



■ **Figure 4** How to predict the color of a node v based on knowledge about the states of nodes in other components. For the sake of illustration, here we are assuming that v belongs to some component which is all white in step t_1 and that the component nearest to v on which a black node appears is V_i . We also suppose that $B_i = \{u_1, u_3\}$ and $\text{dist}(v, u_1) \ll \text{dist}(u_1, u_3) = \text{diam}(V_i)$. In time step t_1 the situation is as in (a). Since $B'_i = \{u_3\}$ is not empty, we must treat V_i as potentially having black nodes since the first time step. We see the states of u_1 and u_3 in t_1 and determine that $\alpha(v) = \text{dist}(v, u_1) + 1 = \text{dist}(v, u_2)$ and $\beta(v) = \text{dist}(v, u_3)$; however, we do not know the color of u_2 since it is inside V_i and we do not query it, so we have to treat it as a potentially black node (denoted by a question mark). After $\alpha(v) - 1$ steps (b) we know that v must still be white (denoted by an exclamation mark) since the closest node to it that is possibly black in step t_1 is the node u_2 . After $\alpha(v)$ steps (c) we are no longer certain about the color of v . After $\text{diam}(V_i)$ steps (d) we know that u_1 must be black, but we still cannot say anything about v . Finally after $\beta(v)$ steps (e) we are sure that v has turned black at the latest since u_3 was black in t_1 .