Electronic Colloquium on Computational Complexity, Report No. 34 (2021)

Robust Self-Ordering versus Local Self-Ordering

Oded Goldreich[*]

March 9, 2021

Abstract

We study two notions that refers to asymmetric graphs, which we view as graphs having a unique ordering that can be reconstructed by looking at an unlabeled version of the graph.

A *local self-ordering* procedure for a graph G is given oracle access to an arbitrary isomorphic copy of G, denoted G', and a vertex v in G', and is required to identify the name (or location) of v in G, while making few (i.e., polylogarithmically many) queries to G'. A graph $G = (V, E)$ is *robustly self-ordered* if the size of the symmetric difference between E and the edge-set of the graph obtained by permuting V using any permutation $\pi : V \to V$ is proportional to the number of non-fixed-points of π and to the maximal degree of G; that is, any permutation of the vertices that displaces t vertices must "displace" $\Omega(t \cdot d)$ edges, where d is the maximal degree of the graph.

We consider the relation between these two notions in two regimes: The bounded-degree graph regime, where oracle access to a graph means oracle access to its incidence function, and the dense graph regime, where oracle access to the graph means access to its adjacency predicate.

We show that, *in the bounded-degree regime*, robustly self-ordering and local self-ordering are almost orthogonal; that is, even extremely strong versions of one notion do not imply very weak versions of the other notion. Specifically, we present very efficient local self-ordering procedures for graphs that possess derangements that are almost automorphisms (i.e., a single incidence is violated). One the other hand, we show robustly self-ordered graphs having no local self-ordering procedures even when allowing a number of queries that is a square root of the graph's size.

In the dense graph regime, local self-ordering procedures are shown to yield a quantitatively weaker version of the robust self-ordering condition, in which the said proportion is off by a factor that is related to the query complexity of the local self-ordering procedure. Furthermore, we show that this quantitatively loss is inherent. On the other hand, we show how to transform any robustly self-ordered graph into one having a local self-ordering procedure, while preserving the robustness condition. Combined with prior work, this yields explicit constructions of graphs that are both robustly and locally self-ordered, and an application to property testing.

[*] Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel. E-mail: oded.goldreich@weizmann.ac.il. Partially supported by the Israel Science Foundation (grant No. 1041/18).

ISSN 1433-8092

Contents

1 Introduction

In this work we study two notions that refers to asymmetric graphs, which we view as graphs having a unique ordering that can be reconstructed by looking at an unlabeled version of the graph (equiv., by exploring any graph that is isomorphic to it). These notions, called *robust self-ordering* and *local self-ordering*, were introduced by Goldreich and Wigderson [4, 5], where the focus was on constructions and applications. The current work is devoted to studying the relation between these two notions.

Robustly self-ordered graphs. Loosely speaking, a *robustly self-ordered graph* is as far as possible from having even approximate automorphisms; that is, for any t, any permutation of the vertices that displaces t vertices must "displace" $\Omega(t \cdot d)$ edges, where d is an upper-bound on the degree of the graph. In other words, a graph $G = ([n], E)$ is called robustly self-ordered if for every permutation $\pi : [n] \to [n]$ it holds that G and $\pi(G)$ differ on $\Omega(d) \cdot |\{v \in [n] : \pi(v) \neq v\}|$ vertex-pairs, where $\pi(G) = ([n], \{\{\pi(u), \pi(v)\} : \{u, v\} \in E\})$; that is, if π has t non-fixed-points, then the symmetric difference between the graphs is $\Omega(t \cdot d)$.

We shall consider two regimes (or types of graphs). In the regime of *bounded-degree graphs* being robustly self-ordered means that the symmetric difference is at least a constant fraction of the number of non-fixed-points. In contrast, in the regime of *dense graphs*, where the degree bound is the trivial one (i.e., the number of vertices), being robustly self-ordered means that the symmetric difference is at least a $\Omega(n)$ times the number of non-fixed-points, where n denotes the number of vertices in the graph.

The two regimes also differ in what it means to have oracle access to the graph. In the regime of *bounded-degree graphs* this means having oracle access to the incidence function of the graph; that is, the query (v, i) is answered with the i^{th} neighbor of v (and by a special symbol in case v has less than than i neighbors). In the regime of *dense graphs* oracle access to the graph means access to its adjacency predicate; that is, the query (u, v) is answered 1 if u is adjacent to v in the graph, and is answered 0 otherwise. (Indeed, these oracles are the ones considered in the bounded-degree and dense graph models of the property testing literature (cf., e.g., [2, Chap. 8-9]).)[1]

Local self-ordering procedures. Loosely speaking, a *local self-ordering procedure for G*, is given oracle access to an arbitrary isomorphic copy of G, denoted G', and a vertex v in G', and is required to identify the name (or location) of v in G, while making few queries to G'. That is, a local self-ordering procedure for G is a randomized algorithm that, on input $v \in [n]$ and oracle access to $G' = \pi(G)$, makes poly$(\log n)$ queries to G' and outputs $\pi^{-1}(v)$ (with probability at least $1 - n^{-c}$, for any desired constant c). We stress that π is *a priori* unknown to this procedure, but indeed $\pi^{-1}(v)$ is partial information (about π) obtained by the procedure.

1.1 The question and a brief summary of the answers

It seems that the two aforementioned notions may be related. They both carry an air of resiliency of the self-ordering phenomenon; that it, they both require the graph to be self-ordered (a.k.a asymmetric) in a very strong sense. In the case of robust self-ordering this strong sense is expressed in global terms (i.e., the size of the symmetric difference between a graph and a graph in which

[1] The definitions of robustness in the two regimes also fits the definition of distance in these two property testing models.

some vertices are displaced), and in the case of local self-ordering the criteria is local (i.e., a unique neighborhood identifying each vertex). This begs the following question:

> *Does robust self-ordering imply local self-ordering and is it implied by it?*
>
> That is, does any robustly self-ordered graph have a locally self-ordering procedure and does the existence of such a procedure imply that the graph is robustly self-ordered?

Loosely speaking, we show that in the *bounded-degree graph regime* the answers are extremely negative: Robustly self-ordered graphs may have no local self-ordering procedure, even when allowing such procedure to use $o(\sqrt{n})$ rather than $\text{poly}(\log n)$ many queries, whereas local self-ordering is possible in graphs that have derangements that are extremely close to being automorphisms (i.e., a single incidence is violated).

In the *dense graph regime* local self-ordering procedures only for graphs that satisfy a quantitatively weaker version of the robust self-ordering condition, but these graphs need not be robustly self-ordered in the full-fledged sense. We also show how to transform any robustly self-ordered graph into one having a local self-ordering procedure, while preserving the robustness condition. The question of whether all robustly self-ordered graphs have local self-ordering procedures is left opened.

We interpret these results as saying that the global and local versions of the vague concept of resiliency of self-ordering are not as tightly related as one could have thought. Especially in the dense graph regime, this interpretation is quantitative in nature, just as the results on which it is based. Hence, a more clear discussion of the question does require more quantitative definitions than the ones outline above. We provide such definitions next, and will re-visit the stated results later.

1.2 Quantitative definitions

Although the following definitions are presented in terms of individual graphs, we view these graphs as belonging to some infinite sets (or families) of graphs and presents the various quantities as function of the size of the graph (i.e., the number of vertices in it). For example, when we talk of dense graphs we envision a family of graphs such the maximum degrees of vertices in an n-vertex graph is $\Omega(n)$. In contrast, when we talk of bounded-degree graphs we envision a family of graphs such the maximum degrees of vertices in an n-vertex graph is bounded by a constant d, which is independent of n. With these preliminaries in place, we turn to the actual definitions.

Definition 1.1 (robustly self-ordered graphs): *For a function $\gamma : \mathbb{N} \to \mathbb{R}$, we say that graph $G = (V, E)$ is γ-robustly self-ordered if for every permutation $\pi : V \to V$ it holds that*

$$|E \triangle \{\{\pi(u), \pi(v)\} : \{u, v\} \in E\}| \ \geq\ \gamma(|V|) \cdot |\{v \in V : \pi(v) \neq v\}|. \tag{1}$$

where $S \triangle T$ denotes the symmetric difference between the sets S and T.

Recall that in the bounded-degree regime, we say that a family is robustly self-ordered if γ is lower-bounded by a positive constant (i.e., $\gamma(n) = \Omega(1)$). In contrast, in the dense graph regime, we say that a family is robustly self-ordered if γ is lower-bounded by a positive linear function (i.e., $\gamma(n) = \Omega(n)$). Note that if $G = (V, E)$ has maximum degree d, then for every permutation $\pi : V \to V$ the size of symmetric difference between G and $\pi(G) = (V, \{\{\pi(u), \pi(v)\} : \{u, v\} \in E\})$ is upper-bounded by $d \cdot |\{v \in V : \pi(i) \neq i\}|$.

Definition 1.2 (local self-ordering procedures): *For a function $q : \mathbb{N} \to \mathbb{N}$, a q-*query local self-ordering procedure *for a self-ordered graph $G = ([n], E)$ is a randomized oracle machine that, given a vertex v in any graph $G' = (V', E')$ that is isomorphic to G and oracle access to G', makes at most $q(n)$ queries, and outputs, with probability at least $2/3$, the vertex that corresponds to v in G; that is, it outputs $\phi(v) \in [n]$ for the unique bijection $\phi : V' \to [n]$ such that $\phi(G') = G$* (i.e., the unique isomorphism of G' to G)*.*

The definition outlined at the beginning of the introduction mandates that q is a polylogarithmic function, but we may consider arbitrary sub-linear functions. Recall that in the bounded-degree graph regime, with degree bound d, oracle access to G' means oracle access to the incidence function of G' (i.e., the function $g' : V \times [d] \to V \cup \{\perp\}$ such that $g'(v, i)$ is the i^{th} neighbor of v in G' and $g'(v, i) = \perp$ if v has less than than i neighbors). In contrast, in the regime of dense graphs, oracle access to G' means access to its adjacency predicate (i.e., the function $g' : V \times V \to \{0, 1\}$ such that $g'(u, v) = 1$ if and only if u is adjacent to v in G').

We stress that, in contrast to [4, Def. 4.6], Definition 1.2 does not place restrictions on the time complexity of the procedure. We shall, however, refer to the time complexity of local self-ordering procedures when proving that even efficient ones do not have certain implications, making such negative results stronger.

1.3 The bounded-degree graph regime

Recall that in the bounded-degree graph regime we seek graphs that are $\Omega(1)$-robustly self-ordered, and that oracle access to a graph (as provided to a local self-ordering procedure) means oracle access to its incidence function. We show that in this regime robustly self-ordering and local self-ordering are almost orthogonal.

Theorem 1.3 (robustness versus locality in the bounded-degree regime):

1. *There exist explicit n-vertex bounded-degree graphs that are not $(3/n)$-robustly self-ordered, but have $O(\log n)$-time deterministic local self-ordering procedures.*
2. *There exist n-vertex bounded-degree graphs that are $\Omega(1)$-robustly self-ordered, but have no $o(\sqrt{n})$-query local self-ordering procedure.*

Part 1 (proved in Theorem 3.1) asserts that even an extremely strong notion of local self-ordering (i.e., deterministic procedures of logarithmic time complexity), fails to imply an extremely weak notion of robust self-ordering (i.e., anything above merely being self-ordered). Note that, query complexity $O(\frac{\log n}{\log\log n})$ is the absolute minimum for any local self-ordering procedure, because the actual information contents of the answers to q queries is an unlabelled graph with q edges, whereas the number of such unlabelled graphs is $\exp(O(q \log q))$. On the other hand, local self-ordering a n-vertex graph requires that the graph is asymmetric (i.e., $\Omega(1/n)$-robustly self-ordered), and so ruling out $3/n$-robustness is the extreme, because any non-trivial permutation must displace at least one edge (contributing two units to the symmetric difference).[2]

Part 2 (proved in Theorem 3.2) is also very strong: It asserts that full-fledged robustly self-ordered graphs do not imply the existence of very weak local self-ordering procedure that are

[2] Indeed, $3/n$ stands for any quantity larger than $2/n$.

allowed $o(\sqrt{n})$ queries (rather than $\text{poly}(\log n)$ queries). This result is proved using random $O(1)$-regular graphs. An explicit construction that supports a $n^{\Omega(1)}$ query lower bound is presented in Proposition 3.4.

The proofs of Parts 1 and 2 (see Theorems 3.1 and 3.2, resp.) clarify that, in the bounded-degree graph regime, the robust self-ordering feature has little to do with local self-ordering procedures. What matters for the latter is the ability to locate oneself based on exploring a small portion of the graph, where in bounded-degree graphs such an exploration is truly local. In particular, a directed and positional (binary) tree (used in the proof of Theorem 3.1) is the archetypical structure that supports such a localization. We note that these directed and colored edges can be implemented by constant-size gadgets (which are either trees (having vertices of different degrees) or have constant girth). Needless to say, these graphs may not be robustly self-ordered. In contrast, a regular graph of large girth (which may be robustly self-ordered) hinders any attempt at such localization, since the neighborhoods look identical anywhere. In fact, different vertex degrees and short cycles are the only ways one can gather information about one's location in the graph by exploring the neighborhood. (We stress that the foregoing discussion refers to the bounded-degree regime, and makes little sense in the dense graph regime.)

A glance at the techniques. Part 2 of Theorem 1.3 is proved by reducing the problem of establishing lower bounds on the query complexity of local self-ordering procedures for a *regular* graph G to analyzing the probability of closing a (simple) cycle in a random (non-backtracking) walk on G. Specifically, the (query complexity) lower bound is inversely proportional to the square root of the said probability. We note that non-backtracking random walks, introduced in [1], are harder to analyze than standard random walks. Nevertheless, we show that the said probability in a random regular graph is inversely proportional to the size of the graph. It also follows that, for any regular graph, the query complexity of local self-ordering is exponential in its girth. This fact is used in the proof of Proposition 3.4.

Remark 1.4 (on the difference between the two constructions of robustly self-ordered graphs that were presented in [4, Part I]): *The proof of Proposition 3.4 actually shows that the construction of $\Omega(1)$-robustly self-ordered n-vertex graphs presented in [4, Sec. 3] does not have $n^{o(1)}$-query locally self-ordering procedures. In contrast, the construction of $\Omega(1)$-robustly self-ordered n-vertex graphs presented in [4, Sec. 4] has a* $\text{poly}(\log n)$*-time locally self-ordering procedure [4, Thm. 4.7]. This articulates a fundamental difference between these two constructions.*

1.4 The dense graph regime

Recall that in the dense graph regime we seek n-vertex graphs that are $\Omega(n)$-robustly self-ordered, and that oracle access to a graph (as provided to a local self-ordering procedure) means oracle access to its adjacency predicate. We show that, in this regime, local self-ordering procedures do not imply the full-fledged notion of robust self-ordering, but do imply a quantitatively weaker version of it.

Theorem 1.5 (from locality to robustness in the dense graph regime):

1. *For every $\ell(n) \in [\Omega(\log n), o(n)]$, there exist n-vertex graphs that have a $O(\ell(n)^2)$-query local self-ordering procedure, but are not $\omega(n/\ell(n))$-robustly self-ordered. Furthermore, the procedure is non-adaptive and relies on inspecting a subgraph induced by $O(\ell(n))$ vertices.*

2. *Every n-vertex graph that has a $q(n)$-query local self-ordering procedure is $\Omega(n/q(n)^2)$-robustly self-ordered. Furthermore, if the procedure is non-adaptive, then the graph is $\Omega(n/q(n))$-robustly self-ordered.*

Recall that a procedure is called non-adaptive if it determines all its queries beforehand (i.e., based on its explicit input and internal coin tosses, but independently of the answers to previous queries).

Theorem 1.5 asserts that there is a quantitative relation between the query complexity of the local self-ordering procedure and the robustness parameter of the graph: As one may expect, the robustness parameter is inversely related to the query complexity of the local self-ordering procedure. Specifically, Part 2 asserts that the robustness is at least inversely proportional to the non-adaptive query complexity, and Part 1 assert that this relation is rather tight (i.e., the robustness may be inversely proportional to a square root of the non-adaptive query complexity).

Part 1 of Theorem 1.5 suggests that graphs having local self-ordering procedures may be easier to construct than $\Omega(n)$-robustly self-ordered n-vertex graphs. The proof of Theorem 2.5 supports this feeling because it presents a relatively simple construction of n-vertex graphs coupled with efficient local self-ordering procedures. These graphs are not $\Omega(n)$-robustly self-ordered, and the construction is significantly simpler than the construction of the latter (presented in [4, Sec. 8]).

The foregoing seems to suggest that robustly self-ordered is a stronger condition than having a local self-ordering procedure. However, we do not know whether this "suggestion" is true (i.e., whether every $\Omega(n)$-robustly self-ordering graph has a $\text{poly}(\log n)$-query local self-ordering procedure).

Open Problem 1.6 (does $\Omega(n)$-robustness imply $\text{poly}(\log n)$-locality): *Is it the case that any $\Omega(n)$-robustly self-ordered n-vertex graph has a* $\text{poly}(\log n)$*-query local self-ordering procedure?*

A positive answer to Problem 1.6 would have allowed us to establish [5, Thm. 1.2–1.4] while using efficiently recognizable graph properties (by using the explicit construction of robustly self-ordered graphs provided in [4, Thm. 8.10]).[3] Fortunately, these positive corollaries also follow from an efficient transformation of $\Omega(n)$-robustly self-ordered n-vertex graph to ones that have $\text{poly}(\log n)$-query local self-ordering procedure. We do provide such a transformation.

Theorem 1.7 (from robustness to locality in the dense graph regime): *There exists a polynomial-time transformation of $\Omega(n)$-robustly self-ordered n-vertex graphs to $\Omega(n)$-robustly self-ordered $O(n)$-vertex graphs that have* $\text{poly}(\log n)$*-time local self-ordering procedures. Furthermore, the transformation is local in the sense that the adjacency predicate of the resulting graph can be computed by a* $\text{poly}(\log n)$*-time reduction to the adjacency predicate of the input graph.*

(Indeed, the time bounds on these local procedures imply corresponding query complexity bounds.) Combining Theorem 1.7 with [4, Thm. 8.10] (or even the weaker [4, Thm. 1.4]), we obtain graphs that are both robustly and locally self-ordered.

Corollary 1.8 (explicit constructions obtaining linear robustness and polylogarithmic locality): *There exists an explicit construction of n-vertex graphs that are $\Omega(n)$-robustly self-ordered and have local self-ordering procedures that run in* $\text{poly}(\log n)$*-time. Furthermore, the adjacency predicate of these graphs can be computed in* $\text{poly}(\log n)$*-time.*

[3] Doing the same for [5, Thm. 1.5] would have required more significant modifications. In particular, the current proof of [5, Thm. 1.5] uses a non-explicit two-source extractor (see [5, Clm. 5.3]).

As stated above, substituting the non-explicit constructions used in the proofs of [5, Thm. 1.2–1.4] by the (strongly) explicit construction of Corollary 1.8, we obtain an explicit version of [5, Thm. 1.4] (which generalizes [5, Thm. 1.2&1.3]).

Corollary 1.9 (explicit separation between adaptive and non-adaptive testers in the dense graph model): *There exists a polynomial-time recognizable graph property that is testable by an adaptive oracle machine that make $O(\epsilon^{-1} \cdot \sqrt{n} \cdot \mathrm{poly}(\log n))$ queries but testing it non-adaptively requires $\Omega(n)$ queries, where n denotes the number of vertices in the graph. Furthermore, for every functions $f, g : \mathbb{N} \to \mathbb{N}$ such that $f(n) \leq \sqrt{n}$ and $g(m) \leq m$, there exists a polynomial-time recognizable graph property Π that satisfies the following two conditions:*

1. *There exists a general* (i.e., adaptive) *tester of Π that runs in time $O(\epsilon^{-1} \cdot f(n) \cdot \mathrm{poly}(\log n))$, and any tester for Π must make $\Omega(f(n))$ queries.*
2. *Any non-adaptive tester of Π must make $\Omega(g(f(n)) \cdot f(n))$ queries, and there exists such a tester that runs in time $O(\epsilon^{-2} \cdot g(f(n)) \cdot f(n) \cdot \mathrm{poly}(\log n))$.*

Actually, the same holds with respect to the more general result of [5, Thm. 1.7], which we avoid stating here. We stress that Corollary 1.9 provides an upper bound on the computational complexity of the testers, whereas the results in [5] only upper-bound the query complexity. However, the query complexity of the testers asserted by Corollary 1.9 is a $\mathrm{poly}(\log n)$ factor larger than the bounds stated in [5, Thm. 1.2–1.4].

A glance at the techniques. With the exception of Part 2 of Theorem 2.3, all results in this regime are proved by constructions of adequate graphs. The common theme in these constructions is combining several graphs that are "locally distinguishable" and locating the vertices within each part of the combined graph based on their adjacencies in some (but not all) of the parts. The simpler case occurs in the proof of Part 1 of Theorem 2.3, where we combine three (non-explicit) robustly self-ordered graphs that are each locally self-ordered. More sophisticated constructions appear in the proofs of Theorems 2.5 and 1.7, where we combine logarithmically many parts and locate vertices in each part based on their adjacencies in some of the other parts.

1.5 The story: From the initial motivation to a general study

Our initial motivation was proving Corollary 1.9, which asserts explicit graph properties that match results proved in [5] using non-explicit graph properties. Specifically, the results proved in [5, Thm. 1.2–1.4] assert various separation between adaptive and non-adaptive testers in the dense graph model. The properties used there were non-explicit, because they relied on $\Omega(n)$-robustly self-ordered n-vertex graphs that have $\mathrm{poly}(\log n)$-query local self-ordering procedures. At the time, such graphs were only known to exist (via a probabilistic argument [5, Cor. 2.7]), but no explicit construction was known. In contrast, explicit constructions of $\Omega(n)$-robustly self-ordered n-vertex graphs were given in [4, Thm. 1.4], but they were not known to have a $\mathrm{poly}(\log n)$-query local self-ordering procedures.

This gave rise to Problem 1.6 (i.e., does every $\Omega(n)$-robustly self-ordered n-vertex graph have a $\mathrm{poly}(\log n)$-query local self-ordering procedure). While we did not resolve Problem 1.6, the transformation provided by Theorem 1.7 is good enough for the original motivation, since it efficiently transforms the graphs of [4, Thm. 1.4] into ones that are both linearly robust and have local self-ordering procedures of polylogarithmic query complexity.

A natural question that arose was whether the converse implication holds; that is, does the existence of a $\text{poly}(\log n)$-query local self-ordering procedure for a dense n-vertex graph imply that the graph is $\Omega(n)$-robustly self-ordered? This question was answered negatively in Part 1 of Theorem 1.5, whereas Part 2 asserts that such a graph must be $\Omega(n/\text{poly}(\log n)))$-robustly self-ordered.

While the foregoing refers to the dense graph regime, a comparative study of the situation in the bounded-degree graph regime seemed begging. Note that in the latter regime explicit constructions of n-vertex graphs that are $\Omega(1)$-robustly self-ordered and have $\text{poly}(\log n)$-query local self-ordering procedures were known [4, Thm. 4.7].[4] Nevertheless, it is interesting to see whether phenomena that occur in one regime also occur in the other regime, and Theorem 1.3 indicates that this is not the case with respect to the current question (since in the bounded-degree graph regime robustly self-ordering and local self-ordering are almost orthogonal).

1.6 Organization

The next two sections can be read independently of one another: Section 2 studies the dense graph regime, whereas Section 3 is devoted to the study of the bounded-degree graph regime.

The dense regime. Section 2 starts with the presentation of a strong notion of local self-ordering, captured by the notion of a reliable locator (Definition 2.1) and poses the question of whether this is actually a real strengthening (Problem 2.2). In Section 2.1 we prove a rather tight quantitative implication from local self-ordering to robust self-ordering, establishing Theorem 1.5. In Section 2.2 we present a relatively simple construction of dense graphs with an efficient local self-ordering procedures (Theorem 2.5). In Section 2.3, we present a transformation of robustly self-ordered graph into ones having local self-ordering procedures, establishing Theorem 1.7.

The bounded-degree graph regime. In Section 3 we prove Theorem 1.3, which asserts that – in the bounded-degree graph regime – robust self-ordering and local self-ordering are almost orthogonal. Part 1 of the theorem is proved in Section 3.1 by observing that directed and positional (binary) trees have extremely simple local self-ordering procedures. On the other hand, Part 2 of Theorem 1.3 is proved in Section 3.2 by proving that regular graphs have no local self-ordering procedure of complexity q if the probability that a random (non-backtracking) walk on them closes a (simple) cycle with probability $o(1/q^2)$.

2 The Dense Graph Regime

Recall that in the dense graph regime we seek n-vertex graphs that are $\Omega(n)$-robustly self-ordered, and that oracle access to a graph (as provided to a local self-ordering procedure) means oracle access to its adjacency predicate.

It seems that, in the current regime, robustly self-ordering and locally self-ordering are related. On the one hand, the fact that an n-vertex graph is $\Omega(n)$-robustly self-ordered implies that the

[4] Interestingly, our study uncovers a fundamental different between this construction and an alternative construction presented in [4, Sec. 3] (see Remark 1.4).

neighborhoods of its n vertices are pairwise far apart, and so random samples of a $O(\log n)$-vertices would yield n different adjacency-patterns. Furthermore, *if* such a random sample induces a subgraph that is not isomorphic to any subgraph induces by a different set, *which is a big IF*, then we get a locally self-ordering procedure.

On the other hand, the fact that an n-vertex graph has a $\text{poly}(\log n)$-query local self-ordering procedure implies that a random set of $\text{poly}(\log n)$ vertices induces a self-ordered subgraph (w.h.p.).[5] This seems to suggest that the graph is $\Omega(n/\text{poly}(\log n))$-robustly self-ordered, which is indeed the case (see the proof of Theorem 2.4). This establishes Part 2 of Theorem 1.5, but we first prove Part 1, which assert that a graph having such a local self-ordering procedure may not be $\omega(n/\log n)$-robustly self-ordered.

Actually, we shall present a stronger result that asserts that even a strong notion of local self-ordering does not imply robust self-ordering (see Theorem 2.3). Specifically, we refers to the notion of *reliable locators*, introduced in [5]. Loosely speaking, a set S of the vertices of G is called a reliable locator if the subgraph of G induced by S is self-ordered and unique in G, and every other vertex has a unique signature with respect to S (i.e., its sequence of adjacencies with S is unique).

Definition 2.1 (reliable locator of a self-ordered graph [5, Def. 2.5]): *A set of vertices $S \subset [n]$ is called a* reliable locator *of a graph $G = ([n], E)$ if the following two conditions hold*

1. *The subgraph of G induced by S is self-ordered and is not isomorphic to any other induced subgraph of G.*

2. *For every $v \in [n] \setminus S$, the adjacencies of v with S uniquely determine v; that is, for every $u \neq v$ in $[n] \setminus S$ there exists $s \in S$ such that $\{u, s\} \in E$ if and only if $\{v, s\} \notin E$.*

Note that if almost all ℓ-sized subsets in a graph are reliable locators, then this graph has a $O(\ell^2)$-query local self-ordering procedure, which selects a random ℓ-set of vertices S and inspects the subgraph induced by $S \cup \{v\}$, where v denotes the input vertex.

We note that, although one is tempted to think that a $q(n)$-query local self-ordering procedure for an n-vertex graph yields an abundance of reliable locators (via error reduction)[6] it is not clear if this is the case. Indeed, we ask

Open Problem 2.2 (do local self-ordering procedures yield reliable locators?) *Does the existence of local self-ordering procedure of polylogarithmic query complexity imply that almost all polylogarithmically sized setrs are reliable locators?*

2.1 Proof of Theorem 1.5

As stated above, we first prove a strong version of Part 1 of Theorem 1.5.

Theorem 2.3 (local self-ordering does not imply linearly-robust self-ordering): *There exist n-vertex graphs in which all but a $1/\text{poly}(n)$ fraction of the $O(\log n)$-vertex subsets are reliable locators,*

[5] Assume towards the contradiction that the subgraph induced by a random set of s vertices is symmetric with probability p. Letting $\pi = \pi_S$ be any non-trivial automorphism of the subgraph induced by the set S, and consider executions of a procedure that on input v, where v is a non-fixed-point of S, queries pairs in $\binom{S}{2}$. It follows that any $(s-1)$-query procedure fails to locate a random vertex v with probability at least p/s.

[6] Error reduction does imply that almost all $O(q(n) \cdot \log n)$-long sequences of vertices can be used to localize all vertices, but this claim refers to the adjacencies among vertices in a sequence, not to the subgraph induced by a set.

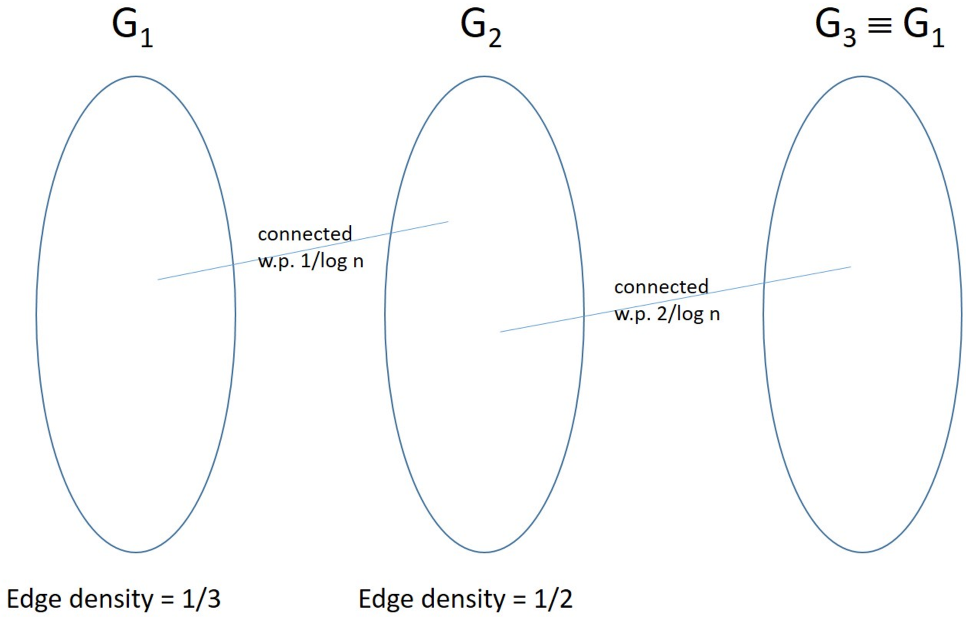

Figure 1: The construction demonstrating Theorem 2.3.

but the graph is not $(n/\log_2 n)$-robustly self-ordered. Furthermore, each vertex in the graph has degree $\Theta(n)$.

Theorem 2.3 can be generalized by replacing $O(\log n)$ with any $\ell = \Omega(\log n)$. That is, *for every $\ell \in [\Omega(\log n), o(n)]$, there exist n-vertex graphs in which all but a* $\exp(-\Omega(\ell))$ *fraction of the ℓ-vertex subsets are reliable locators, but the graph is not $\omega(n/\ell)$-robustly self-ordered.*[7]

Proof: We consider n-vertex graphs that consists of three dense $n/3$-vertex graphs that are connected by random bipartite graphs of low edge density. Specifically, for any constant $p \in (0,1)$, let $\mathcal{G}(k,p)$ be the Erdos–Renyi random k-vertex graph in which each pair of vertices is connected with probability p independently of all other vertex-pairs. Then, for $k = n/3$, we consider a distribution on n-vertex graphs that is generated as follows (see Figure 1).

- Pick $G_1 = ([k], E_1)$ uniformly at random from $\mathcal{G}(k, 1/3)$, and $G_2 = (\{k+1, ..., 2k\}, E_2)$ uniformly at random from $\mathcal{G}(k, 1/2)$. Let $G_3 = (\{2k+1, ..., 3k\}, E_3)$ be an isomorphic copy of G_1, obtained by letting $E_3 = \{\{2k+u, 2k+v\} : \{u,v\} \in E_1\}$.
- Connect G_1 to G_2 by a random bipartite graph in which each edge is included with probability $1/\log_2 n$, independently of all other choices.
- Connect G_3 to G_2 by a bipartite graph in which each edge is included with probability $2/\log_2 n$, independently of all other choices.

Clearly, with overwhelmingly high probability, the resulting graph G is not $(n/\log_2 n)$-robustly self-ordered. To see this, consider a bijection π that switches the vertices of G_1 with the corresponding

[7] This can be seen by replacing all occurrences of $\log_2 n$ in the following proof with $\ell/O(1)$.

vertices of G_3. The symmetric difference between G and $\pi(G)$ is due solely to the edges between G_2 and the other G_i's, but the expected number of these edges is $3k^2/\log_2 n$, whereas the number of non-fixed-points of π is $2k$ (and $\frac{3k^2/\log_2 n}{2k} = \frac{n}{2\log_2 n}$).

On the other hand, we show that all but $1/\text{poly}(n)$ fraction of the $O(\log n)$-subsets of vertices of G are reliable locators. Let S be a random set of $\ell = O(\log n)$ vertices in G, and let S_i denote its intersection with the vertices of G_i. Then, with probability at least $1 - \exp(-\Omega(\ell))$, the following conditions hold:

1. Each S_i has size approximately $\ell/3$.

   (This condition will be used to establish all subsequent ones.)

2. Each vertex of S_1 (resp., S_3) has approximately $\ell/9$ neighbors in S, whereas each vertex of S_2 has approximately $\ell/6$ neighbors in S.

   (In all cases, almost all of the neighbors of vertex $v \in S$ in S reside in the same part S_i as v (i.e., in the same G_i), whereas contribution of the bipartite graphs is at most $O(\ell/\log n)$.)

3. The number of edges between S_1 and S_2 is approximately $\ell^2/9\log_2 n$, whereas the number of edges between S_3 and S_2 is approximately $2\ell^2/9\log_2 n$.

4. Each vertex of G_i has $\Omega(\ell)$ neighbors in S_i, but $o(\ell)$ neighbors in $S \setminus S_i$.

   (Note that here we use a union bound over all n vertices

5. Each S_i constitutes a reliable-locator for G_i.

   This follows by a generalization of [5, Thm. 2.6]. Specifically, while the original assertion refers to $\mathcal{G}(k, 1/2)$, the generalization to $\mathcal{G}(k, p)$ for any constant $p \in (0, 1)$ is quite straightforward (cf. [5, Clm. 5.5]).

Condition 2 allows to distinguish $S' = S_1 \cup S_3$ from S_2, whereas Conditions 3–4 allows to distinguish S_1 from S_3. Specifically, once the set S' is identified, we determine its bi-partition into (S_1, S_3) by considering the subgraph of G induced by S and using Condition 4. (This does not determine which part of the bi-partition is S_1.) Then, by Condition 3, we tell which of the two parts is S_1 and which is S_3. Combining Conditions 4 and 5, we conclude that S is a reliable-locator for G, where Condition 4 allows to determine to which G_i each vertex belongs (and Condition 5 allow to locate it within this G_i). ■

Proving Part 2 of Theorem 1.5. While local self-ordering procedures do not imply linear-robustness, they do imply a sublinear level of robust self-ordering.

Theorem 2.4 (local self-ordering implies weak robust self-ordering): *Suppose that $G = ([n], E)$ has a local self-ordering procedure that uses $q(n)$ non-adaptive queries and errs with probability at most $1/3$. Then, G is $\Omega(n/q(n))$-robustly self-ordered.*

It follows that if G has a local self-ordering procedure that uses $q(n)$ general (adaptive) queries, then it is $\Omega(n/q(n)^2)$-robustly self-ordered, since q adaptive queries can be emulated by $2q^2$ non-adaptive ones (cf. [3]).

Proof: To gain some intuition, consider a permutation π that switches two vertices of G and keeps the rest intact. We claim that in this case the symmetric difference between G and $\pi(G)$ is $\Omega(n/q(n))$. This is shown by first showing that, without loss of generality, a $q(n)$-query local self-ordering procedure queries the graph on adjacencies among the input vertex and $2q(n)$ random vertices. (The argument, which appears below, is similar to the one in [3, Sec. 4.2].) Next, we note that if the symmetric difference between G and $\pi(G)$ is $o(n/q(n))$, then such a procedure cannot distinguish the two vertices that are switched by π, since distinguishing mandates hitting a pair of vertices that is in the symmetric difference. This argument can be extended to the case that π has $o(n/q(n))$ non-fixed-points. In the remaining case, a different but similar argument applies. We now detail the actual proof.

We first detail and prove the simplifying assumption made above. Fixing a non-adaptive local self-ordering procedure for G, we claim that, on input $v \in [n]$, without loss of generality, this procedure makes queries to pairs of the form (s, s') where s is uniformly distributed in $[n]$ and s' is either uniformly distributed in $[n]$ or equals v. This can be seen by considering an algorithm that, on input $v \in [n]$, selects uniformly a permutation $\phi : [n] \to [n]$ such that $\phi(v) = v$, and replaces the query (u, w) by $(\phi(u), \phi(w))$. Hence, the resulting algorithm is of the right form (i.e., its queries are uniformly distributed either in $\binom{[n]\setminus\{v\}}{2}$ or in $\{v\} \times ([n] \setminus \{v\})$), and its correctness follows by observing that when given access to a graph G', it emulates the execution of the original algorithm on the graph $\phi(G')$.

Hence, we may consider an algorithm, denoted A, that on input $v \in [n]$, selects $\ell = 2q(n)$ vertices, denoted $s_1, ..., s_\ell$, uniformly in $[n] \setminus \{v\}$, makes $q(n)$ queries, where each of these queries has either the form (s_i, s_j) or the form (s_i, v), where $i, j \in [n]$. Furthermore, without loss of generality, for some $Q \in \binom{[\ell]}{2}$ that is independent of v the set of queries made by A equals $Q_{\overline{s},v} \stackrel{\rm def}{=} \{(s_i, s_j) : (i, j) \in Q\} \cup \{(s_i, v) : i \in [\ell]\}$, where $\overline{s} = (s_1, ..., s_\ell)$. We may also assume, without loss of generality, that the randomness of A has the form $(s_1, ..., s_\ell, \omega)$ and that A's final output depends only on the answers A has obtained for its queries (i.e., $Q_{\overline{s},v}$), and ω (but is independent of v and the s_i's themselves).[8] Hence, when given oracle access to the graph G, the output of A on input v and randomness $\overline{r} = (s_1, ..., s_\ell, \omega)$, denoted $A^G_{\overline{r}}(v)$, is determined by ω and the sequence of answers provided to the ordered set of queries $\{(s_i, s_j) : (i, j) \in Q\} \cup \{(s_i, v) : i \in [\ell]\}$ (where the ordering is determined by a fixed ordering of Q). We let $A^G(v)$ denote the random variable $A^G_{\overline{r}}(v)$ such that $\overline{r}$ is uniformly distributed (in $\binom{[n]\setminus\{v\}}{\ell} \times \Omega$, for some adequate Ω).

Towards analysing the robust self-ordering of G, we consider an arbitrary bijection $\pi : [n] \to [n]$. Let $T = \{v \in [n] : \pi(v) \neq v\}$ denote the set of non-fixed-points of π, and let $I \subseteq T$ be a set of size at least $|T|/3$ such that $I \cap \pi(I) = \emptyset$. Then, recalling that A is correct on each input, with probability

[8] This is also justified by an argument of the foregoing spirit (which is also adapted from [3, Sec. 4.2]). Here we take an opposite view and consider the behavior of the algorithm that we emulate. Specifically, our starting point is an algorithm A' that on input v' and randomness $(s'_1, ..., s'_\ell, \omega')$, queries its oracle on vertex-pairs in $\{v', s'_1, ..., s'_\ell\}$, and decides according to the answers and its input and randomness. We constructs an algorithm A that, on input v and oracle access to G', selects $\phi \in {\rm Sym}_n$ and ω' uniformly, invokes A' on input $\phi(v)$ and randomness $(\phi(s_1), ..., \phi(s_\ell), \omega')$, answers query (u, w) by querying G' on $(\phi^{-1}(u), \phi^{-1}(w))$, and outputs the output it obtains from A'. Hence, the output of A' is computed according to (part of) the subgraph of G' induced by $(v, s_1, ..., s_\ell)$ and based on input $\phi(v)$ and randomness $(\phi(s_1), ..., \phi(s_\ell), \omega')$, which means that this output is independent of v and $(s_1, ..., s_\ell)$. Formally, on input v and randomness $(s_1, ..., s_n, \omega)$, algorithm A lets $\omega = (u, r_1, ..., r_\ell, \omega')$, where $\{u, r_1, ..., r_\ell\}$ is uniformly distributed in $\binom{n]}{\ell+1}$, and invokes A' on input u and randomness $(r_1, ..., r_\ell, \omega')$. It then answers the query (r_i, r_j) (resp., (r_i, u)) by querying G' on (s_i, s_j) (resp., (s_i, v)), and outputs the output provided by A'. Hence, A emulates an execution of A' on input $\phi(v)$ and oracle $\phi(G')$ such that $\phi(v) = u$ and $\phi(s_i) = r_i$.

at least 2/3, it follows that, for every $v \in I$, it holds that

$$\Pr[A^G(v) \neq A^G(\pi(v))] \;\geq\; \Pr[A^G(v) = v] - \Pr[A^G(\pi(v)) = v] \;\geq\; 1/3, \tag{2}$$

where the foregoing holds regardless of the dependency between the two random invocations of A in the first probabilistic expression. Recalling that $A^G_{\overline{r}}(x)$ denote the output of A^G on input x and randomness $\overline{r} = (s_1, ..., s_\ell, \omega)$, and letting $\mathcal{S}_v \stackrel{\rm def}{=} \binom{[n]\setminus\{v\}}{\ell}$, we can write Eq. (2) as

$$\Pr_{(\overline{s},\omega)\in\mathcal{S}_v\times\Omega}[A^G_{\overline{s},\omega}(v) \neq A^G_{\pi(\overline{s}),\omega}(\pi(v))] \geq 1/3, \tag{3}$$

since $\pi(\overline{s}) = (\pi(s_1), ..., \pi(s_\ell))$ is a uniformly distributed (ordered) ℓ-subset of $[n] \setminus \{\pi(v)\}$. Letting $\chi : [n]^2 \to \{0, 1\}$ denote the adjacency predicate of the graph G (i.e., $\chi(u, v) = 1$ if and only if $\{u, v\}$ is an edge of G), the latter probability bound (i.e., Eq. (3)) implies

$$\Pr_{\overline{s}=\{s_1,...,s_\ell\}\in\mathcal{S}_v}\left[\begin{array}{l} \exists i\in[\ell] \text{ s.t. } \chi(s_i, v) \neq \chi(\pi(s_i), \pi(v)) \\ \vee \;\; \exists (i,j)\in Q \text{ s.t. } \chi(s_i, s_j) \neq \chi(\pi(s_i), \pi(s_j)) \end{array}\right] \;\geq\; 1/3, \tag{4}$$

since $A_{\overline{s},\omega}(v) = A_{\pi(\overline{s}),\omega}(\pi(v))$ holds if the same answers were provided to all corresponding queries.[9] We now consider two cases.

Case 1: The first event in Eq. (4) ***is likely.*** That is,

$$\Pr_{\{s_1,...,s_\ell\}\in\binom{[n]\setminus\{v\}}{\ell}}[\exists i\in[\ell] \text{ s.t. } \chi(v, s_i) \neq \chi(\pi(v), \pi(s_i)) > 0.1.$$

It follows that $\Pr_{s\in[n]\setminus\{v\}}[\chi(v, s) \neq \chi(\pi(v), \pi(s))] > 0.1/\ell$. Since this holds for every $v \in I$, whereas $I \cap \pi(I) = \emptyset$, we get

$$\begin{aligned} |\{(v, s)\in[n]^2 : \chi(v, s) \neq \chi(\pi(v), \pi(s))\}| \;\;&\geq\;\; \sum_{v\in I} |\{s\in[n] : \chi(v, s) \neq \chi(\pi(v), \pi(s))\}| \\ &>\;\; \frac{0.1 \cdot (n-1)}{\ell} \cdot |I|, \end{aligned}$$

which is $\Omega(n/q(n)) \cdot |T|$, since $|I| \geq |T|/3$ and $\ell = 2q(n)$.

Case 2: The second event in Eq. (4) ***is likely.*** That is,

$$\Pr_{\{s_1,...,s_\ell\}\in\binom{[n]\setminus\{v\}}{\ell}}[\exists (i,j)\in Q \text{ s.t. } \chi(s_i, s_j) \neq \chi(\pi(s_i), \pi(s_j))] > 0.1.$$

It follows that $\Pr_{\{r,s\}\in\binom{[n]\setminus\{v\}}{2}}[\chi(r, s) \neq \chi(\pi(r), \pi(s))] > 0.1/|Q|$, which implies

$$|\{(r, s)\in[n]^2 : \chi(r, s) \neq \chi(\pi(r), \pi(s))\}| \;>\; 0.1 \cdot (n-1)^2/|Q|.$$

This is definitely $\Omega(n/q(n)) \cdot |T|$.

It follows that G is $\Omega(n/q(n))$-robustly self-ordered. ■

[9] Recall that $A_{\overline{s},\omega}(v)$ makes the queries $\{(s_i, v) : i \in [\ell]\} \cup \{(s_i, s_j) : (i, j) \in Q\}$, whereas $A_{\pi(\overline{s}),\omega}(\pi(v))$ makes the queries $\{(\pi(s_i), \pi(v)) : i\in[\ell]\} \cup \{(\pi(s_i), \pi(s_j)) : (i, j)\in Q\}$. Furthermore, both rule based solely on the answers and ω.

2.2 Explicit construction of graphs with local self-ordering procedures

In light of Theorem 2.3, it is natural to ask whether it is easier to obtain locally self-ordering procedures for n-vertex graphs that are not $\Omega(n)$-robustly self-ordered. This seems to be the case, as demonstrated next: The point is that the construction presented in the following proof is much simpler than the known construction of $\Omega(n)$-robustly self-ordered, which relies on non-malleable two-source extractors (cf. [4, Sec. 8]).

Theorem 2.5 (explicit construction with an efficient procedure): *There exists an efficient construction of a family of graphs that have a local self-ordering procedure. Furthermore, the graphs are locally constructable* (i.e., the adjacency of two vertices in the graph can be determined in polynomial-time) *and the local self-ordering procedure runs in* $\text{poly}(\log n)$*-time, where n denotes the number of vertices in the graph. Moreover, local self-ordering procedure is based on reliable locators. On the other hand, the graph is not $\omega(n/\log^2 n)$-robustly self-ordered and each vertex in the graph has degree $\Theta(n/\log n)$.*

Using Theorem 2.4, it follows that these n-vertex graphs are $(n/\text{poly}(\log n))$-robustly self-ordered. We note that Theorem 2.5 does not supersede Theorem 2.3, because the vertex degrees are linear in Theorem 2.3 and the quantitative relation is tighter there.[10]

Proof: The graph consists of logarithmically many cliques of noticeably different sizes that are connected in a way that reflects the location of the vertices in each clique: The (approximate) degree of each vertex identifies to which clique it belongs, whereas its exact location in this clique is indicated by the identity of the cliques in which it has neighbors. Indeed, each vertex has relatively few neighbors in other cliques, but sufficiently many such neighbors so to enable the localization procedure.

Specifically, For $\ell = O(\log n)$ and $m = \Theta(n/\ell^2)$, we consider a graph that consists of ℓ cliques of different sizes that are connected by bipartite graphs of edge-density $1/\text{poly}(\ell)$. The i^{th} clique has $(\ell + i) \cdot m = \Theta(n/\ell)$ vertices, and each of its vertices is connected to at most $m/2$ vertices in other cliques. The identity of the cliques to which each vertex is connected encode its location; that is, we encode each vertex by an $\ell/3$-bit long string and connect it to $\ell/3$ of the (subsequent) cliques. Details follow (see Figure 2).

***The cliques*:** For each $i \in \mathbb{Z}_\ell = \{0, 1, ..., \ell - 1\}$, let C_i be an $(\ell + i) \cdot m$-vertex clique, with vertex-set $V_i = \{\langle i, j\rangle : j \in [(\ell + i) \cdot m]\}$.

Let $V = \bigcup_{i \in \mathbb{Z}_\ell} V_i$, and note that $n = |V| = \Theta(\ell^2 \cdot m) < 2^{\ell/3}$.

***Cells in cliques*:** For each $i \in \mathbb{Z}_\ell$, we partition V_i into $2\ell^2 + 2i\ell$ cells, each of size $m' = m/2\ell$; that is, for every $j \in \mathbb{Z}_{2\ell^2 + 2i\cdot\ell}$, let $V_{i,j} = \{\langle i, r\rangle : r \in [j \cdot m' + 1, ..., j \cdot m' + m']\}$.

(Indeed, $|V_i| = (2\ell^2 + 2i\ell) \cdot m' = (\ell + i) \cdot m$.)

***Connecting the cells*:** Using any (efficiently computable and invertible) injective map $M : V \rightarrow \{0, 1\}^{\ell/3}$ as an encoding, for every i, j (as above) and $s \in [\ell/3]$, we connect each vertex

[10] Recall that Theorem 2.3 refers to n-vertex graphs in which each vertex has degree $\Theta(n)$. Also recall that Theorem 2.3 asserts an abundance of $O(\log n)$-vertex subsets that are reliable locators in an n-vertex graph that is not $(n/\log_2 n)$-robustly self-ordered. In contrast, the proof of Theorem 2.5 yields an abundance of $O(\log n)^5$-vertex subsets that are reliable locators in an n-vertex graph that is not $\omega(n/\log^2 n)$-robustly self-ordered.

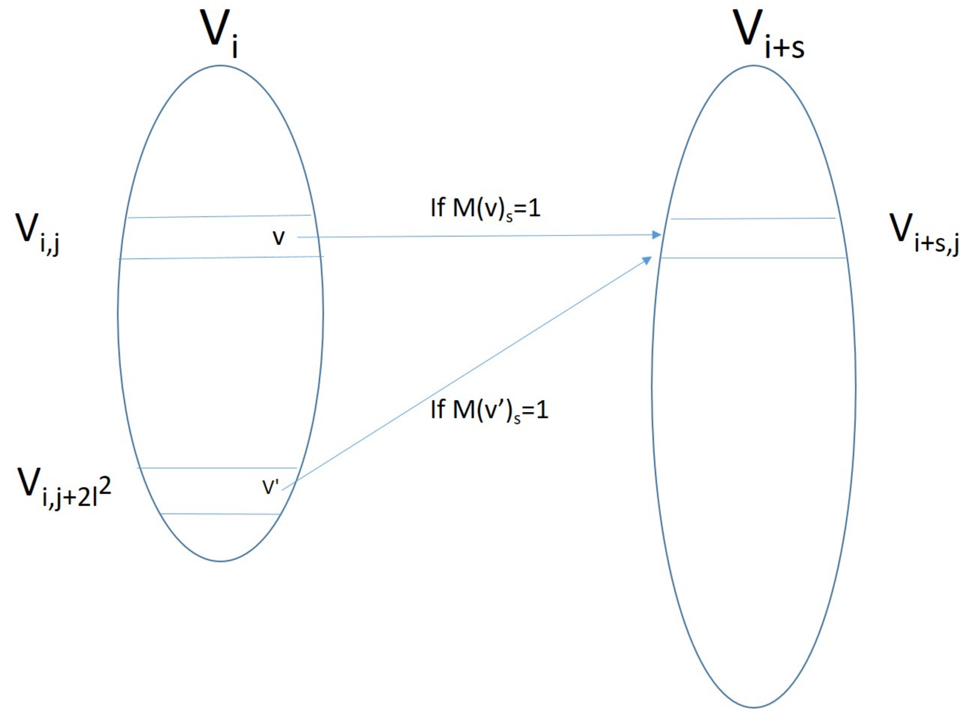

Figure 2: Detail in the construction demonstrating Theorem 2.5.

$v = \langle i, r\rangle \in V_{i,j}$ to all vertices in $V_{i+s \bmod \ell, j \bmod 2\ell^2}$ if and only if the s^{th} bit of $M(v)$ is 1; that is, $v \in V_{i,j}$ is connected to $u \in V_{i',j'}$ if and only if the following three conditions hold: (1) $i' \in \{i+s \bmod \ell : s \in [\ell/3]\}$, (2) $j' = j \bmod 2\ell^2$, (3) $M(v)_s = 1$.

Hence, for $j \in \mathbb{Z}_{2\ell^2}$, vertices in $V_{i,j}$ are only connected to vertices in either V_i or $V_{i+s \bmod \ell, j}$ or $V_{i-s \bmod \ell, j} \cup V_{i-s \bmod \ell, j+2\ell^2}$ such that $s \in [\ell/3]$. As for $j \in \mathbb{Z}_{2\ell^2+2i\ell} \setminus \mathbb{Z}_{2\ell^2}$, vertices in $V_{i,j}$ are only connected to vertices in either V_i or $V_{i+s \bmod \ell, j-2\ell^2}$ such that $s \in [\ell/3]$. See Figure 2. Note that, for every i, j and $s \in [\ell/3]$, if a vertex $v \in V_i$ is connected to some vertex in $V' \stackrel{\text{def}}{=} V_{i+s \bmod \ell, j}$, then v is connected to all $m' = \Omega(n/\ell^3)$ vertices in V'.

Envisioning the C_i's as residing on a cycle, the vertices in each C_i are only connected to the $C_{i'}$'s that are at distance at most $s \in [\ell/3]$ from C_i on the cycle, where the edges between V_i and V_{i+s} are confined to $\bigcup_{j \in \mathbb{Z}_{2\ell^2+2i\ell}} (V_{i,j} \times V_{i+s, j \bmod 2\ell^2})$.

Denoting the resulting graph by $G = (V, E)$, we observe that vertices of the different cliques can be easily told apart based on their degree, because vertices in C_i have degree at least $(\ell + i) \cdot m - 1$ and at most

$$(\ell + i) \cdot m + \frac{\ell}{3} \cdot \frac{m}{2\ell} + \frac{\ell}{3} \cdot 2 \cdot \frac{m}{2\ell} = (\ell + i + 0.5) \cdot m,$$

where the first term is due to the edges of C_i, and the second (resp., third) term is due to the edges that connect C_i and C_{i+s} (resp., C_i and C_{i-s}) for $s \in [\ell/3]$.

A vertex v in a graph $G' = (V', E')$ that is isomorphic to G is located (in G) by taking a sample S of $\text{poly}(\log n)$ random vertices in G', and querying the subgraph of G' induces by $S \cup \{v\}$. Using the degrees of the vertices in this induced subgraph, we identifying the clique to which each of these vertices belongs, and using the adjacency of v we obtain the encoding of the corresponding vertex of G under M, which yields its identity. That is we decide that v corresponds to vertex $\langle i,r\rangle$ of G if v appears to reside in a clique of size $(\ell+i+0.25\pm 0.25)\cdot m$ and for every $s\in[\ell/3]$ the vertex v is connected to a vertex that seems to resides a clique of size $(\ell+i+s+0.25\pm 0.25)\cdot m$ if and only if $M(\langle i,r\rangle)_s=1$.

More specifically, approximating the degree of $\mathrm{poly}(\ell)$ many vertices upto $\pm 0.1m/\ell$ (w.h.p.) can be done by using a sample of $\widetilde{O}(\ell^6)$ vertices, since $m=\Theta(n/\ell^2)$, and so the foregoing procedure can be implemented using $\widetilde{O}(\ell^{12})$ queries. A closer look reveals that it suffices to use a sample of $\widetilde{O}(\ell^3)$ vertices in order to hit all the $V_{i,j}$'s (w.h.p.), since $m'=\Theta(n/\ell^3)$, whereas we only need to approximate the degrees of the vertices in this sample upto $\pm 0.1m$ (w.h.p.), which can be done using a sample of size $\widetilde{O}(\ell^4)$. Hence, local self-ordering can be done using $\widetilde{O}(\ell^7)$ non-adaptive queries. (We mention that, w.h.p., a sample of size $O(\ell^5)$ constitutes a reliable locator, where the extra $\widetilde{\Theta}(\ell)$ factor arises from the need to approximate the degree of all n vertices.)

Lastly, we upper-bound the robustness parameter of G by considering a permutation π that switches C_1 and $(\ell+1)\cdot m$ of the vertices of C_2. The contribution of each non-fixed-point to the symmetric difference between G and $\pi(G)$ is smaller than $2m=O(n/\log^2 n)$, since the number of edges that go outside each clique is at most $m/2$ (and the difference in the sizes of C_1 and C_2 is m). The claim follows. ■

2.3 From robustness to locality: Proof of Theorem 1.7

While we do not know the answer to Problem 1.6 (i.e., whether any $\Omega(n)$-robustly self-ordered n-vertex graph has a $\mathrm{poly}(\log n)$-query local self-ordering procedure), we show an efficient transformation of $\Omega(n)$-robustly self-ordered n-vertex graph into ones that are $\Omega(n)$-robustly as well as have a $\mathrm{poly}(\log n)$-query local self-ordering procedure. In fact, we show that random $\mathrm{poly}(\log n)$-sized sets of vertices in the resulting graphs are reliable locators.

Theorem 2.6 (from robustness to robustness plus locality): *There exists an efficient transformation of n-vertex graphs that are $\Omega(n)$-robustly self-ordered to $3n$-vertex graphs that are $\Omega(n)$-robustly self-ordered and have locally self-ordering procedures that runs in* $\mathrm{poly}(\log n)$*-time. Furthermore, the transformation is local in the sense that the adjacency predicate of the resulting graph can be computed by a* $\mathrm{poly}(\log n)$*-time reduction to the adjacency predicate of the input graph. Moreover, the localization is performed based on the subgraph induced by a random* $\mathrm{poly}(\log n)$*-sized set of vertices.*

Proof: The construction borrows some elements from the proof of Theorem 2.5. In particular, we use the idea of locating a vertex according to its adjacency with large subsets in a fixed partition of the vertex set (e.g., large cliques). Specifically, we shall combine any n-vertex $\Omega(n)$-robustly self-ordered graph G with $\Theta(\log n)$ locally distinguishable $\Theta(n/\log n)$-vertex graphs that will be used to locate vertices in G, where the vertices in the smaller graphs will be located either via these graphs or via the vertices of G. In addition, we connect the various graphs such that the robust self-ordering of G induces robust self-ordering of the larger resulting graph, denoted G'. The latter issue imply that we cannot use relatively sparse connections as in the proof of Theorem 2.5, which will complicate our construction and analysis.

More specifically, the graph G' consists of G and $\Theta(\log n)$ many $\Theta(n/\log n)$-vertex cliques. The cliques are distinguishable by virtue of having a noticeably different number of vertices, and they will be used to locate vertices of G via assigning each vertex v of G a different codeword $C(v)$ that determines to which cliques v is connected such that v is connected to all vertices of the i^{th} clique if $C(v)_i = 1$ and is connected to none of these vertices otherwise. The robust self-ordering of G' will follow from the robust self-ordering of G and the edges connecting G and the rest of G'. Since the edges between G and the rest of G' are used in two different ways, we partition the rest of G' to two parts and connect them to G in different ways. Details follow.

Our construction of $G' = (V', E')$ uses a few ingredients. First, we use the n-vertex graph $G = ([n], E)$ that is guaranteed by the hypothesis. Let $\gamma > 0$ be a constant such that G is $\gamma \cdot n$-robustly self-ordered (and assume for simplicity that $\gamma < 1/50$). Furthermore, for simplicity, we assume that each vertex in G has degree at least $\gamma \cdot n$.[11] The second ingredient we use is an efficiently computable error correcting code, $C : [n] \to \{0,1\}^\ell$, of constant relative distance, where $\ell = O(\log n)$. That is, the code C has distance $\Omega(\ell)$; for simplicity, we assume that the distance is at least $\ell/10$. In addition, we require C to satisfy the following two conditions, where the constant 0.25 is an arbitrary constant in $(0, 0.5)$:

1. Each codeword of C has Hamming weight $(0.25 \pm o(1)) \cdot \ell$; that is, for every $x \in [n]$, it holds that $|\{i \in [\ell] : C(x)_i = 1\}| = (0.25 \pm o(1)) \cdot \ell$.
2. For each coordinate $i \in [\ell]$, it holds that $\Pr_{x \in [n]}[C(x)_i = 1] = 0.25 \pm o(1)$.

(Such a code can be constructed using the concatenated code paradigm.)[12] We shall also use 2ℓ cliques of different sizes, where these sizes are evenly distributed between $0.9n/\ell$ and $1.1n/\ell$. Specifically, the i^{th} clique will have n_i vertices such that $n_i = (10\ell + (-1)^{i \bmod 2} \cdot \lceil i/2 \rceil) \cdot n/10\ell^2$; hence, $\sum_{i=1}^{\ell} n_i = n = \sum_{i=\ell+1}^{2\ell} n_i$ (assuming ℓ is even). Lastly, for simplicity, we shall first assume that n is close to a power of two, and let $\mathtt{IP}_2(v, u)$ denote the inner-product mod 2 (of the binary representation) of the vertices v and w (which are viewed as elements of $[n]$).

Construction 2.6.1 (the graph $G' = (V', E')$): *The graph G' consists of a copy of $G = ([n], E)$ and 2ℓ cliques, denoted $G_1, ..., G_{2\ell}$, where $G_i = (V_i, E_i)$ a clique such that $|V_i| = n_i$, that are connected as follows* (see Figure 3).

- *Each vertex $v \in [n]$ of G is connected to all vertices in G_i such that $C(v)_i = 1$* (and $i \in [\ell]$).
- *Each vertex v in $\bigcup_{j \in [\ell]} V_j \equiv [n]$ is connected to all vertices in $G_{\ell+i}$ such that $C(v)_i = 1$* (and $i \in [\ell]$).
- *Each vertex v in $\bigcup_{j \in [\ell]} V_{\ell+j} \equiv [n]$ is connected to vertex w in G if and only if $\mathtt{IP}_2(v, w) = 1$.*

[11] By the robustness feature, G may contain at most one vertex of degree smaller than $0.5\gamma n$. We can make this vertex isolated at the cost of decreasing the robustness constant to $\gamma/2$, and leave this vertex isolated in the following construction of G'.

[12] We start with a Reed-Solomon code $C' : [n] \to \mathrm{GF}(p)^p$ such that $p = \ell/\ell'$ where $\ell' = \Theta(\log \ell)$, and note that each coordinate of a random codeword is uniformly distributed in $\mathrm{GF}(p)$. Next, we find an adequate inner-code $C'' : \mathrm{GF}(p) \to \{0,1\}^{\ell'}$ by exhaustive search on a pseudorandom $p \times \ell'$ matrix. Specifically, we use an $\exp(-\ell')$-biased space on $2 \cdot p \cdot \ell'$-bit long strings in order to define $p \times \ell'$ Boolean matrices such that each entry is 1 with probability approximately $1/4$, and test the distance between rows as well as the weight of rows and of columns. In the concatenated code C each codeword is a sequence of C''-codewords, which implies that Condition 1 holds. As for Condition 2, note that each coordinate in a *random* C-codeword corresponds to a coordinate in a *random* C''-codeword. Lastly, the relative distance of C is upper-bounded by the product of the relative distances of C' and C''.

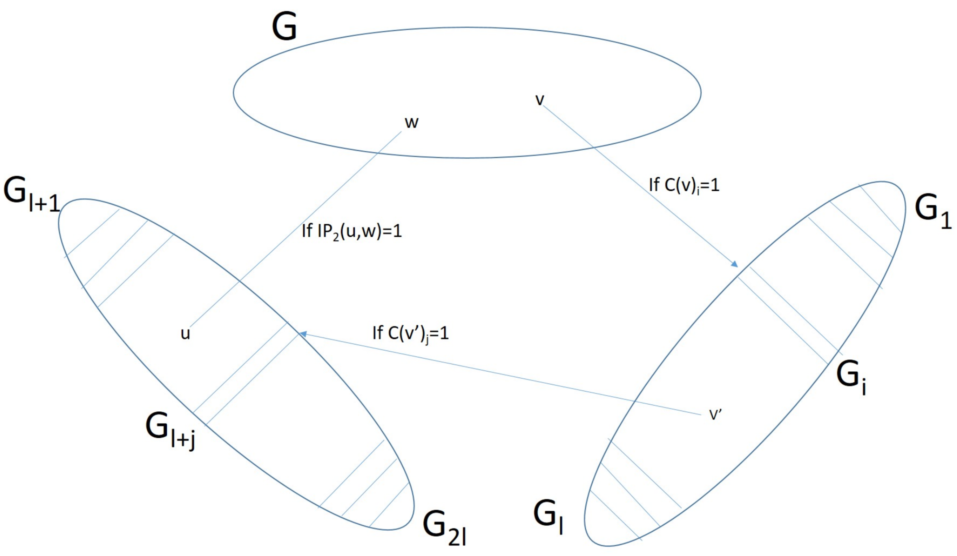

Figure 3: Illustration of Construction 2.6.1.

(Both v and w are viewed as elements of $[n]$.)[13]

Recall that $n_i = (10\ell + (-1)^{i \bmod 2} \cdot \lceil i/2 \rceil) \cdot n/10\ell^2 \in [0.9n/\ell, 1.1n/\ell]$. *We shall refer to a three-way partition of G', calling G the first part, and* $\bigcup_{i\in[\ell]} G_i$ (resp., $\bigcup_{i\in[\ell]} G_{\ell+i}$) *the second* (resp., third) *part. Assuming that ℓ is even, each part has n vertices.*

Note that each vertex in the first part of G' has degree at least $\gamma n + (0.25n - o(n)) + (0.5n - o(n))$, where the first term is due to its neighbors in G and the second (resp., third) term is due to its neighbors in the second (resp., third) part.[14] In contrast, each vertex in the second (resp., third) part has degree $0.5n \pm o(n)$ (resp., $0.75n \pm o(n)$), where the main contribution comes from neighbours in the other two parts (see Footnote 14). Hence, approximating the degree of a vertex in G' determines in which of the three parts of G' it resides. Similarly, for any vertex in the second (resp., third) part of G', we can determine in which G_i it resides by approximating the size of the maximum clique that is induced by its neighbors that reside in the second (resp., third) part. Both observations will be used in the localization process described next.

Claim 2.6.2 (G' is locally self-ordered): *All but* $1/\text{poly}(n)$ *fraction of the* $O(\log^5 n)$*-subsets of* V' *are reliable locators. Furthermore, the location of each vertex can be determined by such a reliable locator in* $\text{poly}(\log n)$*-time.*

Proof: Intuitively, our localization process is anchored in the 2ℓ cliques (i.e., the G_i's). As hinted above, using a small sample of random vertices, it is easy to identify the vertices that reside in G, and identify the clique in which each other vertex resides in (based on the subgraph induced by

[13] Actually, we use two injective mappings $\mu_1 : [n] \to S_n$ and $\mu_2 : \bigcup_{i\in[\ell]} V_{\ell+j} \to S_n$, where S_n is the set of the n lexicographically first strings in $\{0,1\}^{\lceil \log_2(n+1) \rceil} \setminus \{0^{\lceil \log_2(n+1) \rceil}\}$. Hence, $\mathtt{IP}_2(v,w)$ stands for $\mathtt{IP}_2(\mu_2(v), \mu_1(w))$.

[14] Note that the bipartite graph connecting the first and third parts has edge-density approximately 1/2, whereas the other two bipartite graphs (which connect other pairs of parts) have edge-density approximately 1/4.

vertices that are not in G). Using this identification, we can locate the vertices of the first (resp., second) part based on whether or not they neighbor each of the G_i's. The vertices of the third part are located using the locations of their neighbors in the first part. Details follow.

Letting S be a random $O(\log^5 n)$-subset of V', we show how to locate each vertex v of V' (including $v \in S$) based on the subgraph of G' induced by $S \cup \{v\}$. Our localization process proceeds as follow.

1. *Determining which vertices reside in the first part*: We first determine whether v is located in the first part of G', by relying on the degree dichotomy outlined above. Specifically, we rule that v resides in the first part if and only if it has more than $(0.75 + 0.5\gamma) \cdot |S|/3$ neighbors in S.

   Similarly, we determine for each vertex in S whether or not it resides in the first part of G', and let S_1 denote the corresponding set of vertices.

2. *Determining the partition of the rest of S into cliques*: We determine the partition of $S \setminus S_1$ among the 2ℓ cliques by considering the subgraph of G' induced by $S \setminus S_1$. Note that, with high probability (over the choice of S), there exists a unique partition of $S \setminus S_1$ into 2ℓ cliques that are each a subset of one part of G', and that their relative sizes indicate their correspondence to the cliques in G'; that is, with high probability, the set S contains $(1 \pm o(1/\ell^2)) \cdot \frac{n_i}{3n} \cdot |S|$ vertices that reside in the clique G_i.

   Furthermore, the partition can be found in poly$(|S|)$-time, by first identifying to which part of the graph each vertex (in $S \setminus S_1$) belongs, and then using the fact that the vertices in each of these two parts constitute a collection of ℓ isolated cliques.

3. *Locating vertices in the first part*: If v is located in the first part of G', then we determine its exact location in G according to its neighborhood in each of the G_i's of the *second* part. Specifically, denoting the (unknown) location of v in G by x, we determine $C(x)_i$ by checking whether v has a neighbor in G_i (i.e., a neighbor in S that resides in G_i). Note that, with high probability (over the choice of S), it holds that $C(x)_i = 1$ if and only if S contains a vertex that resides in G_i and is connected to v.

   Similarly, we locate each vertex in S_1 (i.e., we determine the exact location of each vertex in S_1).

4. *Locating vertices in the second part*: If v is located in the second part of G', then we determine its exact location in G according to its neighborhood in each of the G_i's of the *third* part. (Indeed, this is done analogously to Step 3.)

5. *Locating vertices in the third part*: If v is located in the third part of G', then we determine its exact location in G according to its neighborhood in the *first* part (or rather its adjacencies to vertices in S_1). We do so while relying on the fact that we have already located all vertices in S_1.

   Specifically, a vertex $s \in S_1$ that is located at vertex w in G yields a linear equation on the location of vertex v in G'. Denoting the latter (unknown) location by x, we get the equation $\mathtt{IP}_2(w, x) = \chi(s, v)$, where $\chi(s, v) = 1$ if and only if s neighbors v (in the subgraph of G' induced by $S \cup \{v\}$). Hence, we get $\Omega(|S|) \gg \ell$ linear equations in x (viewed as an Boolean vector), where the equations are almost uniformly and independently distributed (when S is random).[15] Solving this (full rank) linear system, we obtain x.

Indeed, once the partition of S according to the parts and cliques of G' is determined, the process of localization proceeds from the first part of G' to its second and third parts. That is, we first locate each vertex of the first part based on the *cliques it neighbors* in the second part. Likewise, we locate each vertex of the second part based on the *cliques it neighbors* in the third part. Lastly, we locate each vertex of the third part based on the *location of the vertices of* S_1 *in* G, which means that the localization of a vertex in the third part uses the prior localization of the vertices in S_1.

The foregoing process works provided that S hits each of the cliques (i.e., the G_i's) as well as hits a set of vertices in G whose locations (viewed as $\lceil\log_2(n+1)\rceil$-dimensional vectors) form a basis of the vector space that encodes $[n]$. Hence, with probability at least $1 - 1/\text{poly}(n)$ over the choice of S, we locate each vertex v in G' according to the subgraph induced by $S \cup \{v\}$, which means that S is a reliable locator. ■

Note that, so far (i.e., in the proof of Claim 2.6.2), we did not use the relative distance of C, although we did use the bounds on the weight of its codewords (in order to distinguish the vertices of the different parts according to their degrees). More importantly, the hypothesis that G is $\gamma \cdot n$-robustly self-ordered was not used so far either. Both hypothesis will be used next.

Claim 2.6.3 (G' is $\Omega(n)$-robustly self-ordered): *Recall that C has distance at least 0.1ℓ, and that $G = ([n], E)$ is γn-robustly self-ordered. Then, $G' = (V', E')$ is $\Omega(\gamma n)$-robustly self-ordered.*

Proof: The current proof goes in an opposite direction to the proof of Claim 2.6.2: Whereas the localization process is anchored in the 2ℓ cliques (i.e., the G_i's), the analysis of the robust self-ordering of G' is anchored in the first part (i.e., the robustly self-ordered graph G). Recall that for every permutation μ of the vertices of G', we compare the number of non-fixed-points in μ to the size of the symmetric difference between G' and $\mu(G')$. Intuitively, focusing at permutations that preserve the three-way partition of G', we first observe that the claim follows in the case that many of the non-fixed-points of the permutation reside in the first part, because in this case we can rely on the subgraph of G' induced by the first part (i.e., G). Otherwise, we get a contribution (to the symmetric difference) of vertices in the third part that is due to their different adjacencies with the vertices of the first part. An analogue argument holds for vertices of the second part, based on their different adjacencies with the vertices of the third part. Details follow.

We consider an arbitrary permutation $\mu : V' \to V'$, and its set of non-fixed-points $T = \{x \in V' : \mu(x) \neq x\}$. Recalling that $V' = [n] \cup \bigcup_{i\in[2\ell]} V_i$, we let T' denote the set of vertices that are mapped by μ to a different part of G'; that is,

$$T' \stackrel{\text{def}}{=} \bigcup_{j\in[3]} \{v \in V'_j : \mu(v) \notin V'_j\},$$

where V'_j denotes the j^{th} part of G' (i.e., $V'_1 = [n]$, $V'_2 = \bigcup_{i\in[\ell]} V_i$ and $V'_3 = \bigcup_{i\in[\ell]} V_{\ell+i}$). We consider the following cases.

[15] Here we used the hypothesis that $[n]$ is close to $\{0,1\}^{\lfloor\log_2 n\rceil}$, which implies that the locations of the vertices in S are distributed almost uniformly and identically in $\{0,1\}^{\lfloor\log_2 n\rceil}$.

Case 1: $|T'| > \gamma \cdot |T|/2$**.** In this case, we consider the contribution of the vertices of T' to the symmetric difference between G' and $\mu(G')$. Each such vertex $v \in T'$ contributes $\Omega(n)$ units to the difference, where this contribution is due to the difference between the degree of vertices in the three parts. Specifically, recall that the vertices in the first part have degree at least $\gamma n + (0.75n - o(n))$, whereas the vertices of the second and third parts have degree $0.5n \pm o(n)$ and $0.75n \pm o(n)$, respectively. Hence, in this case, the size of the symmetric difference between G' and $\mu(G')$ is $\Omega(|T| \cdot n)$.

Let $T'' \stackrel{\rm def}{=} \{v \in V_1' : \mu(v) \in V_1' \setminus \{v\}\}$, and note that $T'' \subseteq T \setminus T'$.

Case 2: $|T'| \leq \gamma \cdot |T|/2$ ***and*** $|T''| > |T|/100$**.** In this case, we consider the contribution of the vertices in T'' (which move inside the first part of G' (i.e., G)) to the symmetric difference between G' and $\mu(G')$. Intuitively, by the robust self-ordering of G, each such vertex contributes at least $\gamma \cdot n$ units, but we have to discount the neighbors that moved out of the first part. By the case hypothesis, the number of these potential neighbors (i.e., $|T'|$) is at most $\gamma \cdot n/2$, and so each vertex $v \in T''$ contributes at least $\gamma \cdot n/2$ units. Hence, in this case, the size of the symmetric difference between G' and $\mu(G')$ is $\Omega(|T| \cdot n)$.

Let $T''' \stackrel{\rm def}{=} \{v \in V_3' : \mu(v) \in V_3' \setminus \{v\}\}$, and note that $T'' \subseteq T \setminus (T' \cup T'')$.

Case 3: $|T'| \leq \gamma \cdot |T|/2$ ***and*** $|T''| \leq |T|/100$ ***and*** $|T'''| > |T|/100$**.** In this case, we consider the contribution of the vertices in T''' (which move within the third part of G') to the symmetric difference between G' and $\mu(G')$. The contribution of each such vertex $v \in T'''$ is due to the difference in its adjacencies in the first part; that is, to vertices $w \in V_1'$ that neighbor either v or $\mu(v)$ but not both. Recalling that the adjacency between the third and first parts is determined by $\texttt{IP}_2$, the number of such vertices w equals $|\{w \in V_1' : \texttt{IP}_2(v,w) \neq \texttt{IP}_2(\mu(v),w)\}| \approx n/2$.

As in Case 2, we need to discount vertices w that move out of V_1'; actually, here we need to discard any vertex $w \in V_1'$ that is a non-fixed-point of μ. This means that we should discard at most $|T'| + |T''| < 2n/100$ vertices, where the inequality uses the case hypothesis, which means that each such v contributes $\Omega(n)$ units. Hence, in this case, the size of the symmetric difference between G' and $\mu(G')$ is $\Omega(|T| \cdot n)$.

Case 4: $|T'| \leq \gamma \cdot |T|/2$ ***and*** $|T''| \leq |T|/100$ ***and*** $|T'''| \leq |T|/100$**.** In this case, most vertices in T are vertices v that reside in the second part such that $\mu(v) \neq v$ also resides in the second part. The contribution of each such vertex $v \in T \setminus (T' \cup T'' \cup T''')$ is due to the difference in its adjacencies in the third part; that is, vertices $w \in V_3'$ that neighbor either v or $\mu(v)$ but not both. Recalling that the adjacency between the second and third parts is determined by C applied to vertices of the second part, the number of such vertices w equals

$$\begin{aligned} \sum_{i \in [\ell]} |\{w \in V_{\ell+i} : C(v)_i \neq C(\mu(v))_i\}| &> (0.1 - o(1)) \cdot \ell \cdot \min_{i \in [\ell]}\{n_{\ell+i}\} \\ &> (0.1 - o(1)) \cdot 0.9n, \end{aligned}$$

where the first inequality is due to the distance of C (which we assume to be at least $\ell/10$). Again, we have to discard vertices of V_3' that are non-fixed-points, whereas their number is upper-bounded by $|T' \cup T'''| < 2n/100$. Hence, each such vertex $v \in T \setminus (T' \cup T'' \cup T''')$ contributes $\Omega(n)$ units, and we get a total contribution of $\Omega(|T| \cdot n)$.

Hence, in each of the possible cases, the size of the symmetric difference between G' and $\mu(G')$ is $\Omega(|T| \cdot n) = \Omega(|V'|) \cdot |T|$. ■

Recall that in the foregoing proofs we have assumed that n is approximately a power of two. This assumption was used for a single reason: It guranteed that a uniformly distributed element of $[n]$ has a binary representation that is almost uniformly distributed in $\{0,1\}^{\lfloor \log_2 n \rceil}$, which in turn implies that $\mathtt{IP}_2(u,v)$ is almost unbiased, for every $v \in [n]$ and u that is uniformly distributed in $[n]$. We can obtain the latter effect, for any $n \in \mathbb{N}$, by using ideas as in [4, Rem. 8.9]. Specifically, we replace $\mathtt{IP}_2(u,v)$ by $\mathtt{IP}_2(G(u),G(v))$, where $G : [n] \rightarrow \{0,1\}^{\lceil \log_2(n+1) \rceil} \setminus \{0\}^{\lceil \log_2(n+1) \rceil}$ is a small-biased generator that is injective. Such a generator can be constructed by letting $G(u'u'') = G'(u')u''$, where $|u'| = t = \omega(1)$ such that $\exp(\widetilde{O}(2^t)) \leq \mathrm{poly}(\log n)$, and finding a small-biased generator $G' : [\lceil n/2^{\lceil \log_2(n+1) \rceil - t} \rceil] \rightarrow \{0,1\}^{t+1} \setminus \{0^{t+1}\}$ (which is also injective) by exhaustive search. ■

Digest. Construction 2.6.1 combines a $\Omega(n)$-robustly self-ordered n-vertex graph (i.e., G) with a logarithmic number of $\Theta(n/\log n)$-vertex cliques arranged in two parts. These three parts are connected by bipartite graphs of constant edge-density. The cliques serve as an "anchor" for the locally self-ordering procedure of G': Using the adjacencies of vertices to these cliques, we locate the vertices of the first and second parts of G', whereas the vertices of the third part are located via the already located vertices of the first part. In contrast, the first part (i.e., G) is the anchor for establishing the $\Omega(n)$-robust self-ordering of G': The hypothesis that G is $\Omega(n)$-robustly self-ordered implies that each non-fixed-point of π that resides in the first part contributes $\Omega(n)$ units to the symmetric difference between G' and $\pi(G')$. On the other hand, when almost all vertices in the first part of G' are fixed-points, each non-fixed-point in the third part of G' contributes $\Omega(n)$ units to the symmetric difference, by virtue of their neighbors in the first part. Likewise, if there are few non-fixed-points in the third part, then we lower-bound the size of symmetric difference by considering the non-fixed-points of the second part (and their neighbors in the third part).

Construction 2.6.1 is based on a three-way partition of the vertices, where the original graph G is defined as the first part, and the smaller cliques (i.e., the G_i's) are partitioned among the second and third parts. The reason we use a three-way partition is that we need two different types of edges between G and the rest of the graph. The first type of edges serves for locating the vertices of G according to their adjacencies in the cliques, whereas the second type is used for the analysis of the robust self-ordering of G' (in case the permutation fixes all vertices in G).

3 The Bounded-Degree Graph Regime

Recall that in the bounded-degree graph regime we seek graphs that are $\Omega(1)$-robustly self-ordered, and that oracle access to a graph (as provided to a local self-ordering procedure) means oracle access to its incidence function. We show that in this regime robustly self-ordering and local self-ordering are almost orthogonal; that is, even extremely strong versions of one notion do not imply very weak versions of the other notion.

3.1 Proof of Part 1 of Theorem 1.3

We show that very simple graphs, which are evidently far from being $\Omega(1)$-robustly self-ordered, have simple local self-ordering procedures.

Theorem 3.1 (local self-ordering does not imply robust self-ordering): *There exist explicit n-vertex graphs of maximal degree 3 that have $O(\log n)$-time locally self-ordering deterministic procedures, but are not $(3/n)$-robustly self-ordered.*

This failure is extremely strong. We use the strongest possible notion of local self-ordering (i.e., deterministic procedures of logarithmic[16] time complexity), and rule out an extremely weak notion of robust self-ordering (i.e., anything above merely being self-ordered).

Proof: We first present the construction in terms of directed graphs with edge-coloring. In these terms, the graph is a balanced binary (positioned) tree of depth $\log_2 n$ with edges directed from the root to the leaves, colored left and right, respectively. Specifically, each internal vertex has two children, and the edges are directed from it to its children such that one edge is colored ‘left’ and the other is colored ‘right’.

This (directed and edge-colored) graph can be locally self-ordered by going from the given vertex towards the root (using edges that are directed in the opposite direction), and collecting the edge-colors, which determines the vertex's location. To see that this tree is not $3/n$-robustly self-ordered, consider the permutation that flips the two sides of the tree: This permutation has $n-1$ non-fixed-points, but the corresponding symmetric difference is only 2 (i.e., the two edges incident to the root), which yields a robustness ratio at most $2/(n-1)$.

To transport this example to our model (of undirected graphs with no edge colors), we replaced the two directed and colored edges by two constant-sized gadgets that encode the edge's direction and color. Specifically, an edge from u to v colored ‘left’ is replaced by a 5-path $(u, \mathrm{t}_{u,v}, \mathrm{t'}_{u,v}, \mathrm{h}_{u,v}, \mathrm{h'}_{u,v}, v)$ augmented by a vertex $\mathrm{l}_{u,v}$ that is connected to $\mathrm{t}_{u,v}$, and ditto for color ’right’ except that $\mathrm{l}_{u,v}$ is connected to $\mathrm{t'}_{u,v}$, where ‘t’ stands for tail, ‘h’ for head, and ‘l’ for leaf.

Note that each type of vertex in the resulting tree is easily identified by its constant distance neighborhood (which identifies the gadget in which it resides or those to which it is connected). Note that the permutation that flips the two sides of the resulting tree has only one fixed-point (i.e., the original root), and that the corresponding symmetric difference is only 2 (i.e., the edges connecting the leaves that appear in the two gadgets incident at the root). ■

3.2 Proof of Part 2 of Theorem 1.3

We show that $\Omega(1)$-robust self-oredered graphs may have no local self-ordering procedures of polylogarithmic query complexity. In fact, we prove a stronger statement.

Theorem 3.2 (robust self-ordering does not imply local self-ordering): *There exist n-vertex bounded-degree graphs that are $\Omega(1)$-robustly self-ordered, but do not have local self-ordering procedures of query complexity $o(\sqrt{n})$. Furthermore, this holds for a $1-o(1)$ fraction of the $O(1)$-regular graphs.*

This failure is quite strong. We rule out local self-ordering procedures of quite high query complexity, regardless of their computational complexity.

Proof: The key observation is that, in the case of regular graphs, we may reduce proving lower bound on the query complexity of local self-ordering procedures to proving upper bounds on the

[16] Actually, one can envision a procedure of query complexity $O(\frac{\log n}{\log\log n})$, but this is the absolute minimum. The reason is that the labels obtained by such a procedure are irrelevant, and so the actual information contents of the answers to q queries is an unlabelled graph with q edges, whereas the number of such unlabelled graphs is $\exp(O(q \log q))$.

probability that such a procedure finds a simple cycle. Furthermore, as long as the procedure does not encounter a simple cycle, it is "practically non-adaptive" (see below). Hence, the probability of finding a simple cycle can be upper-bounded in terms of the probability that a *non-backtracking* random walk closes a simple cycle. (A non-backtracking random walk is a walk that at each step chooses uniformly at random one of the neighbors of the current vertex other than the neighbor visited in the previous step [1].) Let us define this notion first.

Definition 3.2.1 (probability of closing a simple cycle): *For a graph $G = ([n], E)$, the* probability of closing a simple cycle in an ℓ-step random walk, *denoted $\mathsf{cc}_\ell(G)$, is the probability that a non-backtracking random walk that starts at a uniformly distributed vertex s reaches s in its $\ell^{\rm th}$ step after visiting $\ell - 1$ distict vertices. The* probability of closing a simple cycle in G, *denoted $\mathsf{cc}(G)$, is $\max_{\ell\in[n]}\{\mathsf{cc}_\ell(G)\}$.*

The choice to take the maximum up to n is rather arbitrary. We could, as well, have taken the maximum up to the query complexity that we care about (or have taken the supremum over $\mathbb{N}$). We mention that $\mathsf{cc}_\ell(G) = 0$ if ℓ is smaller than the girth of G, and $\mathsf{cc}_{\Omega(\log n)}(G)) = (1 \pm o(1))/n$ if G is a regular expander. We shall show (see Lemma 3.2.3) that most $O(1)$-regular n-vertex graphs G satisfy $\mathsf{cc}(G) = O(1/n)$.

Lemma 3.2.2 (main tool): *For $d \geq 3$ and any d-regular graph $G = ([n], E)$, the probability that a q-query procedure succeeds in locating a random vertex in a graph that is isomorphic to G is upper-bounded by $O(q^2 \cdot \max(\mathsf{cc}(G), 1/n))$.*

Proof: We start with an overview of the proof, which refers to an arbitrary q-query algorithm that explores a graph G', which is isomorphic to G, starting at an arbitrary vertex given to it. We may assume, without loss of generality, that this algorithm is deterministic (by fixing the best possible sequence of coin tosses). Furthermore, we observe that the labels of the vertices that appear as queries or as answers to queries are immaterial; all that matters is the equality and inequality relation between these labels.

Next, we consider the event in which an answer to a query equals a vertex that was visited before, where the probability space is uniform over all G' obtained by letting $G' = \pi(G)$. An *uninformative* case is that we queried u for one of its neighbors, received v as an answer, and later queried v and received the answer u. The informative and in fact *surprising* cases are of two types: Either the answer closes a cycle in the subgraph seen so far, or it joins two connected components of that subgraph. We observe that before any surprising event occurs, the subgraph seen so far is a directed forest, where the edges are directed from the queries to the answers. Furthermore, the structure of this forest is determined beforehand, since the answers are practically determined by this hypothesis; that is, the forest represents a sequence of queries that are all answered by vertices that were unvisited till the time of the query.

At this point we make two key observations. The first observation is that the probability of a surprising event is upper-bounded by $p \stackrel{\rm def}{=} \binom{q}{2} \cdot \max(O(\mathsf{cc}(G)), 1/(n-q))$, where the first term is due to answers that close a cycle and the second term is due to answers that merge two trees. The second observation is that if no surprising event occurs, then (w.v.h.p) the algorithm fails to locate the input (i.e., start) vertex. The latter observation relies on the hypothesis that graph is regular, which implies that no information is leaked by a vertex's degree. Needless to say, the precise argument (presented next) involves providing a more precise description of the operation of a generic algorithm and the "exploration forest" defined by it.

The actual proof. To simplify the exposition, we define the incidence queries as providing not only the neighbor u of the queried vertex v but rather also the index of v in the list of neighbors of u; that is, the query (v, j) is answered by (u, k) such that u is the j^{th} neighbor of v, which is the k^{th} neighbor of u. Needless to say, this only makes the algorithm stronger, but it offers the benefit of assuming (w.l.o.g.) that after making the query (v, j) and receiving the answer (u, k), the algorithm never queries (u, k). This corresponds to avoiding traversing an edge in both directions (i.e., uninformative events are avoided). In addition, we assume (also w.l.o.g.) that the algorithm never makes the same query twice.

Next, we note that the labels of the vertices given as answers to the incidence queries are immaterial; all that matters is the equality and inequality relation between these labels. Likewise, the indices of vertices in their neighbor's incidence list do not matter; all that matters is that our queries refer to indices that were not given as answer (i.e., we may query (v, j) only if (v, j) was not provided as an answer to a prior query). Hence, we may replace the query (v, j) by a request for a neighbor of v that was not known already to be a neighbor of v. Lastly, incidence queries regarding a vertex that was not visited so far are replaced by a request for a new vertex followed by an incidence query to it. In light of the above, the algorithm may refer to vertices by the order in which they were first visited.

Hence, the algorithm's queries are of two types: (1) an incidence query of the form "provide a new neighbor of the i^{th} vertex visited so far" (assuming that this is possible)[17], and (2) a request to provide a new vertex (which is defined as visited once obtained). Note that queries of type (1) may be answered by a vertex that was not visited before (and is defined as visited at this time) or by a vertex that was visited before (either via a different vertex or via a request of type (2)). In the latter case we call the answer surprising and note that it either closes a cycle (within a connected component of the subgraph seen before) or causes two connected components (of this subgraph) to merge.

The connected components that we refer to are those in the directed graph defined by the queries and answers, where edges are directed from a query to its answer. This directed graph contains also the input (or start) vertex. Note that if none of the answers is surprising, then this graph is is directed forest.

Assuming that none of answers is surprising, the sequence of queries made by the algorithm is uniquely determined, since all vertices have the same degree and so the question of whether all neighbors of a vertex were visited is predetermined by the queries (under the assumption that no answer is surprising). The i^{th} element in this query-sequence is either a request for a new vertex or a request for a new neighbor of the j^{th} vertex for some $j \in [i]$. In both cases, the answer is defined to be the $i + 1^{\text{st}}$ visited vertex, where *the input vertex is defined as the first visited vertex.* (Recall that the input vertex is the vertex to be located by the algorithm.)

Note that the definition of the sequence of queries has been presented as referring to an actual execution of the algorithm, when the answers are provided by the graph G', and while assuming that none of the answers is surprising. Nevertheless, the same definition applies to a mental experiment in which the algorithm is invokes and the answers are fictitiously emulated such that none of them is surprising. Indeed, the two definitions collide if no surprising answer occurs in the real execution (where the answers are provided by G'). When this condition does not hold in a real execution, it

[17] If the i^{th} vertex was visited from the j^{th} vertex, then the j^{th} vertex is excluded from the list of possible answers. Also excluded are all neighbors previously visited from the i^{th} vertex. Hence, an answer is possible if and only if less than d neighbors were excluded so far.

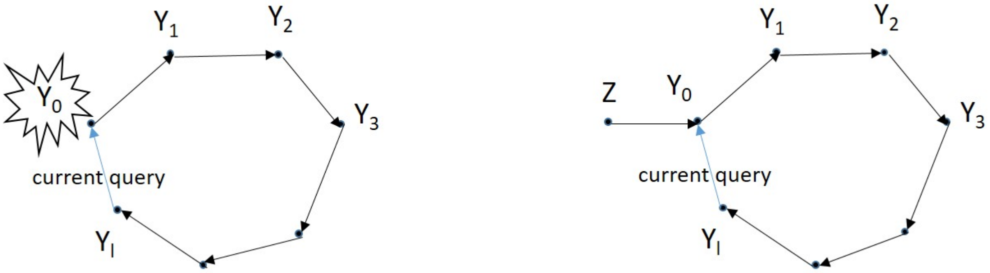

Figure 4: The first two cases in the proof of Lemma 3.2.2.

is the case that the i_1^{th} visited vertex (according to this definition) equals the i_2^{th} visited vertex.

Hence, we shall analyze what happens in a real execution while referring to the forest that is defined by the foregoing mental experiment. This makes sense because our analysis is focused at the case that no surprising event occurs, and as long as this holds both definitions are equivalent. Such a random execution is described by a sequence of random variables $(X_0, ..., X_q)$ such that X_0 is the input vertex (given to the algorithm), and X_i is the answer to the i^{th} query. Specifically, if the i^{th} query is a request for a new vertex, then X_i is uniformly distributed in $[n] \setminus \{X_0, X_1, ..., X_{i-1}\}$, and otherwise X_i in a new neighbor of a previous X_p (i.e., if the i^{th} query is a request for a new neighbor of X_p, where $p \in \{0, 1, ..., i-1\}$, then X_i is uniformly distributed among the unknown neighbors of X_p).[18] A surprising event occurs if $X_i = X_j$ for some $i > j$, and we upper-bound the probability for the occurrence of any surprising event by taking a union bound over all pairs (i, j) and upper-bounding the probability the corresponding event is the first one. That is, for each (i, j) such that $i > j$, we upper-bound the probability that $X_i = X_j$ and $X_{i'} \neq X_{j'}$ for every $i' < i$ and $j' < i'$. Denoting the combined event by $\chi_{i,j}$, we shall prove that $\Pr[\chi_{i,j}] = O(\max(\mathtt{cc}(G), 1/n))$.

Recall that $\chi_{i,j}$ holds only if the i^{th} query is a request for a new neighbor of some vertex X_p such that $p < i$. Furthermore, $(X_0, ..., X_{i-1})$ are distinct. Still we distinguish two cases: The case that X_p and X_j reside in the same tree (where $X_i = X_j$ closes a simple cycle that contains the edge $\{X_p, X_j\}$), and the case that X_p and X_j reside in different trees (where $X_i = X_j$ merges these trees with the edge $\{X_p, X_j\}$).

Starting with the first case, which is more interesting, and letting ℓ denote the distance between X_j and X_p on the tree, we show that in this case $\Pr[\chi_{i,j}] = O(\mathtt{cc}_{\ell+1}(G))$, where the probability space is uniform over all permutations $\pi : [n] \to [n]$ and the algorithm gets its answers from $G' = \pi(G)$. To clarify the exposition, we rename the vertices on the path between X_j and X_p by $Y_0, Y_1, ..., Y_\ell$ such that $Y_k = X_{i_k}$ is the vertex at distance k from $Y_0 = X_j$ on the said path (e.g., $i_0 = j$ and $i_\ell = p$).[19] Note that this path is not necessarily directed from Y_0 to Y_ℓ, and the following subcases will distinguish the two possibilities. Another distinction refers to the question of whether one of the Y_k's is the root of the tree on which all these Y_k's reside. Hence, we have four subcases analyzed next, while recalling that in all subcases the $(\ell+1)$-cycle consists of the foregoing ℓ-path and the edge $\{Y_\ell, Y_0\}$.

1. The vertex Y_0 is a root of the tree in which it resides (i.e., Y_0 is a response to a "new vertex"

[18] That is, X_k is excluded if either X_k was provided as a prior answer to a new neighbor request regarding X_p or X_p was provided as answer to a new neighbor request regarding X_k.

[19] Indeed, the i_k's are distinct.

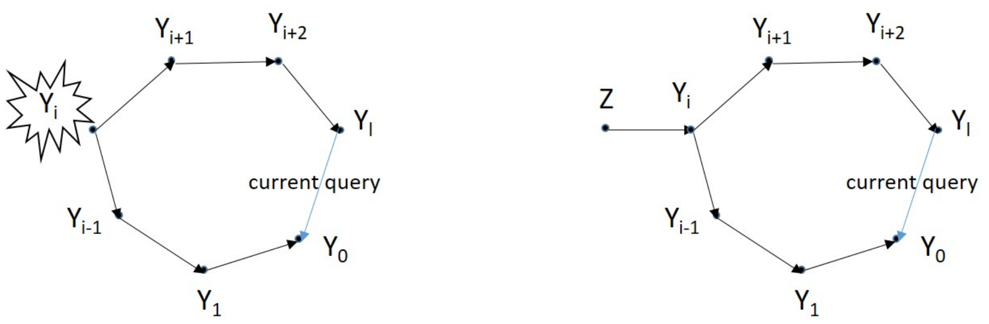

Figure 5: The other two cases in the proof of Lemma 3.2.2 (using $i = 3$ and $\ell = 6$).

request), and each other Y_i is the answer to a ("new neighbor") query regarding Y_{i-1}. In this subcase, a simple directed path of length ℓ leads from Y_0 to Y_ℓ, and the edge from Y_ℓ to Y_0 closes a simple directed $(\ell+1)$-cycle. (See the l.h.s. image in Figure 4.)

The probability that the algorithm traverses this simple $(\ell+1)$-cycle equals the probability that a $(\ell+1)$-step non-backtracking random walk closes a simple cycle in G, which equals $\mathtt{cc}_{\ell+1}(G)$. The foregoing assertion uses the observation that the order in which the algorithm traverses edges of the forest is immaterial, and so we may assume that the path from Y_0 to Y_ℓ is traversed before any other edge is traversed. The latter assumption is justified by the hypothesis that Y_0 is the root of a tree.

2. The vertex Y_0 is not the root of the tree in which it resides (i.e., Y_0 is a response to a "new neighbor" request regarding some Z), and (as in the previous subcase) each other Y_i is the answer to a ("new neighbor") query regarding Y_{i-1}. Hence, as in the previous subcase, we obtain a simple directed $(\ell+1)$-cycle that goes from from Y_0 to Y_ℓ, and then returns to Y_0. (See the r.h.s. image in Figure 4.)

   The probability that the algorithm traverses this simple $(\ell+1)$-cycle equals the probability that a $(\ell+1)$-step non-backtracking random walk closes a *simple* cycle in G *conditioned on the first step not taking a specific neighbor of* Y_0 (i.e., not taking the aforementioned vertex Z). This means that the probability space of the walks taken by the algorithm is cut by a factor of $(d-1)/d$. It follows that in this subcase the probability that a simple $(\ell+1)$-cycle is formed is upper-bounded by $\frac{d}{d-1} \cdot \mathtt{cc}_{\ell+1}(G)$.

3. For some $i \in [\ell]$, the vertex Y_i is a root of the current tree, and the edges are directed from Y_i to Y_ℓ (as in the previous subcases) and from Y_i to Y_0. That is, the path $(Y_0, ..., Y_\ell)$ consists of two directed subpaths, one going from Y_i to $Y_{i+1}, ..., Y_\ell$, and the other going from Y_i to $Y_{i-1}, ..., Y_0$. (Recall that all Y_j's are distinct, and see the l.h.s. image in Figure 5.)

   Like in the second subcase, the current subcase differs from the first subcase in that the probability space of possible walks is cut by a factor of $d/(d-1)$, since it contains two choices for a new neighbors of Y_i and a single choice for each Y_j such that $j \in [\ell-1]$ (rather than containing a single choice for each Y_j such that $j \in \{0, 1, ..., \ell-1\}$). (In other words, the space of possible choices corresponds to $2 \cdot \binom{[d]}{2} \cdot [d-1]^{\ell-2}$ rather than to $[d] \cdot [d-1]^{\ell-1}$ (as in the first subcase).) Hence, again, the probability that a simple $(\ell+1)$-cycle is formed is upper-bounded by $\frac{d}{d-1} \cdot \mathtt{cc}_{\ell+1}(G)$.

4. Lastly, as in the third subcase, for some $i \in [\ell]$, the edges are directed from Y_i to Y_ℓ and from Y_i to Y_0, but Y_i is not a root of the current tree (analogously to the second subcase). (See the r.h.s. image in Figure 5.)

   In this subcase, we make two choices for neighbors of Y_i, but have to avoid a specific neighbor (as in the second subcase). Hence, the space of possible choices corresponds to $2 \cdot \binom{[d-1]}{2} \cdot [d-1]^{\ell-2}$, which means that the probability space (of the non-backtracking random walks) is cut by a factor of $\frac{d(d-1)}{(d-1)(d-2)}$. Hence, in this subcase, the probability that a simple $(\ell+1)$-cycle is formed is upper-bounded by $\frac{d}{d-2} \cdot \mathtt{cc}_{\ell+1}(G)$.

Hence, in all subcases, we got a probability bound of at most $3 \cdot \mathtt{cc}_{\ell+1}(G)$, since $d \geq 3$.

We now turn to the second case, which is that these two vertices (i.e., X_p and X_j) reside in different trees (which are merged by the edges $\{X_p, X_j\}$). The probability that this event occurs (i.e., that a new neighbor request for X_p is answered by X_j) is at most $1/(n-q)$, because there are at least $n-q$ possible locations for the root of the second tree and only one of them yields the said event. It follows that the probability of a surprising event occurring during the q-query execution on G' is at most $\epsilon \overset{\text{def}}{=} \binom{q}{2} \cdot \max(3 \cdot \mathtt{cc}(G), 1/(n-q))$, as claimed.

Lastly, assuming that $\epsilon < 1/2$, we claim that the local self-ordering procedure errs with very high probability (when querying G' that is isomorphic to G), because, *when no surprising events occurs*, little information is revealed on the location of the start (i.e., input) vertex s in G. The key observation is that *the only information that is revealed to the algorithm is the non-occurring of a surprising event.* Hence, fixing $s \in [n]$, for a uniformly distributed permutation $\pi : [n] \to [n]$, we consider the distribution of the location of s in G (i.e., $\pi^{-1}(s)$) given that no surprising event occurs (when exploring $G' = \pi(G)$). Specifically, let $\chi^{G'}(s)$ denote the foregoing surprising event, where $G' = \pi(G)$ is a uniformly distributed isomorphic copy of G. Then, for every $i \in [n]$, it holds that

$$\begin{aligned}
\Pr_\pi[\pi^{-1}(s)=i \,|\, \neg\chi^{\pi(G)}(s)] &= \frac{\Pr_\pi[\neg\chi^{\pi(G)}(s) \,|\, \pi^{-1}(s)=i] \cdot \Pr_\pi[\pi^{-1}(s)=i]}{\Pr_\pi[\neg\chi^{\pi(G)}(s)]} \\
&\leq \frac{\Pr_\pi[\pi^{-1}(s)=i]}{\Pr_\pi[\neg\chi^{\pi(G)}(s)]} \\
&\leq \frac{1/n}{1-\epsilon}
\end{aligned}$$

which is less than $2/n$. This means that $\pi^{-1}(s)$ is hard to predict by making q queries. Specifically, when exploring a random isomorphic copy $G' = \pi(G)$ of G, the probability that a q-query algorithm correctly locates the start vertex is smaller than $\epsilon + (2/n)$. ■

Proving the claim of the theorem. Using Lemma 3.2.2, it suffices to upper-bound the probability of closing a simple cycle in a random d-regular graph.

Lemma 3.2.3 (the cycle closing probability of random graphs): *For any $d \geq 3$ and $t > 1$, for at least $1 - O(1/t^2)$ of the d-regular n-vertex graphs G, it holds that $\mathtt{cc}(G) = O(t/n)$.*

Taking $t = \omega(1)$ and recalling that $1 - o(1)$ fraction of the $O(1)$-regular graphs are $\Omega(1)$-robustly self-ordered (see [4, Thm. 6.1]), the theorem follows.

Proof sketch: Using the fact that, with very high probability, the graph G is an expander, we may assume that $\mathtt{cc}_\ell(G) < 1.01/n$ for all $\ell = \Omega(\log n)$. Hence, we focus on upper-bounding $\mathtt{cc}_\ell(G)$ for $\ell = O(\log n)$.

For every such ℓ, we consider random variables that represent all possible non-backtracking walks of length ℓ on a random d-regular graph $G = ([n], E)$. Such a walk is represented by a start vertex $s \in [n]$ and a sequence of ℓ (non-backtracking) moves $\alpha \in D \stackrel{\rm def}{=} [d] \times [d-1]^{\ell-1}$, where α_i is an index in the list of neighbors of the current vertex that exclude the previous vertex. (That is, the walk reprsented by the pair (s, α) has the form $(v_0 = s, v_1, ..., v_\ell)$, where v_i is the $\alpha_i^{\rm th}$ neighbor of v_i in a list that excludes v_{i-1}.) We stress that selecting a random d-regular graph includes also ordering at random the d edges incident at each vertex.

We define 0-1 random variables $\zeta_{s,\alpha} = \zeta_{s,\alpha}(G)$ such that $\zeta_{s,\alpha} = 1$ if the walk on the random graph G that corresponds to the pair (s, α) returns to s in its last step after visiting $\ell - 1$ different vertices. Note that the average of the $\zeta_{s,\alpha}(G)$'s equals the probability that a random non-backtracking ℓ-step walk on G closes a simple cycle; that is,

$$\mathtt{cc}_\ell(G) = \frac{1}{n \cdot |D|} \cdot \sum_{s \in [n], \alpha \in D} \zeta_{s,\alpha}(G). \tag{5}$$

Our aim is to prove that, for every $t > 1$, the sum (i.e., Eq. (5)) exceeds $O(t/n)$ with probability $O(1/t^2)$, where the probability space is uniform over all d-regular graphs G.

Intuitively, it seems that $\mathrm{E}[\zeta_{s,\alpha}] \approx 1/n$ and that the $\zeta_{s,\alpha}$'s are almost pairwise independent. Indeed, we shall shortly show that this is essentially the case. But first, let us assume that $\mathrm{E}[\zeta_{s,\alpha}] \leq B/n$ and that the sum of the covariances (i.e., $\mathrm{E}[\zeta_{s,\alpha}\zeta_{s',\alpha'}] - \mathrm{E}[\zeta_{s,\alpha}]^2$) of all (distinct) pairs is at most $C \cdot |D|^2$. Then, using Chebyshev's inequality we get

$$\Pr\left[\sum_{s \in [n], \alpha \in D} \zeta_{s,\alpha} > t \cdot |D|\right] < \frac{|D| + C \cdot |D|^2}{(t-B)^2 \cdot |D|^2} = \frac{1}{(t-B)^2} \cdot \left(\frac{1}{d \cdot (d-1)^{\ell-1}} + C\right). \tag{6}$$

Taking the sum over all $\ell \in [O(\log n)]$ and assuming $t > 2B$, we get a probability bound of $O((1 + C \cdot \log n)/t^2)$; that is, $\Pr[\mathtt{cc}_\ell(G) > t/n] = O((1 + C \cdot \log n)/t^2)$. The claim follows once we establish $B = O(1)$ and $C = O(1/\log n)$, since then $\Pr[\mathtt{cc}_\ell(G) > t/n] = O((\exp(-\ell) + O(1/\log n))/t^2)$ for any $t > O(1)$ (where the case of $t \leq O(1)$ holds vacuously).

We start by upper-bounding B; specifically, we prove that $p_\ell \stackrel{\rm def}{=} \mathrm{E}[\zeta_{s,\alpha}] \leq 1/(n - \ell)$. This can be seen by fixing the first $\ell - 1$ steps in the walk that starts at s (and proceeds according to α), which means fixing $\ell - 1$ incidences in the random graph G. Denoting the corresponding vertices by $v_1, ..., v_{\ell-1}$, where $v_0 = s$, we assume that they are distict (or else $\zeta_{s,\alpha} = 0$), and observe that $n - \ell$ vertices were not visited at all and therefore each of them has d free incidences. In contrast, we used one incidence of each v_i, including $v_0 = s$, to get to v_{i+1}, which means that vertex s has only $d - 1$ free incidences. Hence, the probability (over the residual choice of G) that s is chosen in the last step, as the $\alpha_\ell^{\rm th}$ neighbor of $v_{\ell-1}$, is at most $\frac{d-1}{d \cdot (n-\ell)}$, and it follows that $p_\ell \leq \frac{d-1}{d} \cdot \frac{1}{n-\ell}$.

Towards upper-bounding the covariances of all (distinct) variable pairs, we also establish a lower bound on p_ℓ (for $\ell \geq 3$).[20] This is done by noting that the first $\ell - 1$ steps visit distinct vertices with proability $1 - O(\ell/n)$, and taking a closer look at the free incidences. The point is that we used one

[20] Note that $p_1 = p_2 = 0$.

incidence of $v_0 = s$, two incidences of each v_i such that $i \in [\ell - 2]$, and a single incidence of $v_{\ell-1}$. Hence, the number of free incidences is $(n - \ell) \cdot d + 2 \cdot (d - 1) + (\ell - 2) \cdot (d - 2) = n \cdot d - 2 \cdot (\ell - 1)$, and it follows that $p_\ell = (1 - O(\ell/n)) \cdot \frac{d-1}{n \cdot d - 2 \cdot (\ell - 1)} > \frac{d-1}{d} \cdot \frac{1-O(\ell/n)}{n}$.

We now show that, although the $\zeta_{s,\alpha}$'s are not pairwise independent, the sum of their covariances is at most $O(|D|^2 \cdot \ell/n)$; that is, we shall show that $C = O(\ell/n)$. (We comment that a weaker bound of $C = O(\ell)$ is easier to obtain, but it only implies a probability bound of $\text{poly}(\log n)/t^2$).)[21] Using these bounds (i.e., $B = O(1)$ and $C = O(\ell/n)$) in Eq. (6), we get a probability bound of $\frac{O(\exp(-\ell))+o(1/\log n))}{t^2}$, and the claim follows by taking the sum over all $\ell \in [O(\log n)]$.

Hence, it is left to prove the claim regarding the sum of all covariances; that is, the covariances of $\zeta_{s,\alpha}$ and $\zeta_{s',\alpha'}$ for all $s, s' \in [n]$ and $\alpha, \alpha' \in D$ such that $(s, \alpha) \neq (s', \alpha')$. We consider two cases according to the relation between s and s'.

***Case 1:* $s = s'$.** In this case $\alpha \neq \alpha'$ and a crude upper-bound on the covariance of $\zeta_{s,\alpha}$ and $\zeta_{s,\alpha'}$ will suffice, because there are relatively few such pairs. In particular, we upper-bound the covariance of $\zeta_{s,\alpha}$ and $\zeta_{s,\alpha'}$ by $\Pr[\zeta_{s,\alpha} = \zeta_{s,\alpha'} = 1]$, and upper-bound the latter by the product of $\Pr[\zeta_{s,\alpha} = 1]$ and the probability that both walks end at the same vertex (although they take different paths (i.e., $\alpha \neq \alpha'$)). Details follow.

Let γ denote the longest common suffix of α and α', and note that $i \stackrel{\text{def}}{=} \ell - |\gamma| \geq 1$ and $\alpha_i \neq \alpha'_i$. Then, the probability that after i steps the two walks (corresponding to $\zeta_{s,\alpha}$ and $\zeta_{s,\alpha'}$) are at the same vertex is at most $1/(n - 2\ell)$, since in the i^{th} step different incidences are used (regardless of whether or not the walks are at the same vertrex after $i - 1$ steps). This upper bound holds also when conditioning on $\zeta_{s,\alpha} = 1$, since it is derived by fixing the first walk and considering the choice made in the i^{th} step of the second walk. On the other hand, if the two walks are at different vertices after i steps, then with probability at least $1 - ((\ell - i)/(n - 2\ell)$ they will end-up in different vertices (since, as long as they are at different vertices, the next step depends on different incidences). Hence,

$$\Pr[\zeta_{s,\alpha} = \zeta_{s,\alpha'} = 1] \;\leq\; \Pr[\zeta_{s,\alpha} = 1] \cdot \left(\frac{1}{n - 2\ell} + \frac{\ell - i}{n - 2\ell} \right) \;=\; O(\ell/n^2).$$

Observing that there are less than $n \cdot |D|^2$ pairs in this case, the sum of their covariances is $n \cdot |D|^2 \cdot O(\ell/n^2) = O(|D|^2 \cdot \ell/n)$.

***Case 2:* $s \neq s'$.** In this case, we upper-bound the covariance of $\zeta_{s,\alpha}$ and $\zeta_{s',\alpha'}$ more carefully, while relating it to $\Pr[\zeta_{s,\alpha} = 1]^2$. Fixing the first walk, we consider two sub-cases: In the first sub-case the second walk does not intersect the first walk, and the analysis of the second walk is very similar to the analysis of $\Pr[\zeta_{s,\alpha} = 1]$, except that here at most 2ℓ vertices may have less than d free incidences. The complementary sub-case occurs with probability at most $O(\ell^2/n)$, and we can show that the residual walk (after the collision) will hit s' with probability $O(1/n)$. Combining both sub-cases, it follows that in this case $\Pr[\zeta_{s,\alpha} = \zeta_{s',\alpha'} = 1]$ is at most $\Pr[\zeta_{s,\alpha} = 1]^2 + O(\ell^2/n^3)$. Details follow.

[21] Specifically, we (crudly) upper-bound the covariance of $\zeta_{s,\alpha}$ and $\zeta_{s,\alpha'}$ by $\Pr[\zeta_{s,\alpha} = \zeta_{s,\alpha'} = 1]$, and upper-bound the latter by considering two cases. In case $s = s'$, we use a bound of $\Pr[\zeta_{s,\alpha} = 1] = O(1/n)$ and the fact that there are less than $n \cdot |D|^2$ such pairs. In case $s \neq s'$, we upper-bound $\Pr[\zeta_{s,\alpha} = \zeta_{s,\alpha'} = 1]$ by the product of $\Pr[\zeta_{s,\alpha} = 1]$ and the probability that the second walk either hits the first one or ends at s', where each of the latter events occurs with probability $O(\ell/n)$. Hence, the sum of all covariances is upper-bounded by $O(|D|^2) + O(n^2 \cdot |D|^2) \cdot O(\ell/n^2) = O(\ell) \cdot |D|^2$.

Our focus is on upper-bounding $q \stackrel{\rm def}{=} \Pr[\zeta_{s',\alpha'} = 1 \,|\, \zeta_{s,\alpha} = 1]$, and this is done by fixing an arbitrary walk that witnesses $\zeta_{s,\alpha}=1$. As stated above, the probability that the second walk intersects the first walk is $O(\ell^2/n)$, and in this sub-case we consider the intersection point v and argue that the probability that the residual walk that starts at v ends at s' is $O(1/n)$. Hence, the contribution of this sub-case to q is $O(\ell^2/n^2)$.

Our main concern is with the complementary sub-case in which the second walk does not intersect the first walk. In this sub-case the analysis of the second walk is very similar to the analysis of $\Pr[\zeta_{s',\alpha'} = 1]$, except that here at most ℓ additional vertices (i.e., those on the first walk) may have less than d free incidences. Specifically, we may assume that the $\ell - 1^{\rm st}$ vertex in the walk is different from the first one (or else $\zeta_{s',\alpha'} = 0$), and therefore the probability that the walk ends at s' is at most $\frac{d-1}{d} \cdot \frac{1}{n-2\ell}$, since at least $n - 2\ell$ vertices have d free incidences. Hence, in thus subcase we get a probability bound of $p'_\ell \leq \frac{d-1}{d} \cdot \frac{1}{n-2\ell}$. Recalling that $p_\ell > \frac{d-1}{d} \cdot \frac{1-O(\ell/n)}{n}$, we get $p'_\ell \leq \frac{n}{n-O(\ell)} \cdot p_\ell$, which means that the contribution of this sub-case to q is at most $\frac{n}{n-O(\ell)} \cdot p_\ell = (1 - O(\ell/n)) \cdot p_\ell$.

Combining the contribution of both sub-cases, we upper-bound $\Pr[\zeta_{s,\alpha}=\zeta_{s,\alpha'}=1] = p_\ell \cdot q$ by $p_\ell \cdot (O(\ell^2/n^2) + (1 + O(\ell/n)) \cdot p_\ell)$, which is $(1 + O(\ell/n)) \cdot \Pr[\zeta_{s,\alpha}=1]^2$, since $p_\ell = \Omega(1/n) = \omega(\ell^2/n^2)$. Using $\Pr[\zeta_{s,\alpha}=1] = O(1/n)$, it follows that the corresponding covariance is

$$\Pr[\zeta_{s,\alpha}=\zeta_{s,\alpha'}=1] - \Pr[\zeta_{s,\alpha}=1]^2 \;=\; O(\ell/n) \cdot \Pr[\zeta_{s,\alpha}=1]^2 \;=\; O(\ell/n^3).$$

Hence, in this case the sum of all covariances is $n^2 \cdot |D|^2 \cdot O(\ell/n^3) = O(|D|^2\ell/n)$.

Hence, the sum of all covariances is $O(|D|^2 \cdot \ell/n)$, and the claim follows. ■

This completes the proof of the theorem. ■

Digest. The proof of Theorem 3.2 reduces the problem of establishing lower bounds on the query complexity of local self-ordering procedures for a graph G to analyzing the probability of closing a simple cycle in a random walk on G. The definition of this probability referred to non-backtracking random walks (see Definition 3.2.1). This type of random walks, introduced in [1], are harder to analyze than standard random walks, but they fit our application well. Nevertheless, an analogous definition of the probability of forming a simple cycle in a random walk is natural also in the context of other types of random walks (e.g., standard or lazy ones).[22] Turning back to Definition 3.2.1, we observe that the probability of closing a simple cycle decreases exponentially with the girth of the graph. That is –

Proposition 3.3 (cc decreases exponentially with the girth): *For every d-regular graph G of girth g it holds that* $\texttt{cc}(G) \leq (d-1)^{-(g-1)/2}$.

Using Lemma 3.2.2 it follows that any d-regular n-vertex graph that has logarithmic girth does not have local self-ordering procedures of query complexity $O(n^{o(1)})$.

[22] Note that in the latter cases it makes little sense to consider the probability of closing a simple cycle, since these walks are likely to backtrack. On the other hand, one may consider the probability of forming a simple cycle (rather than closing one) also in the case of non-backtracking random walks. However, as noted above, Definition 3.2.1 fits our application best.

Proof: Clearly, $\mathtt{cc}_\ell(G) = 0$ for every $\ell \in [g-1]$. For $\ell > (g-1)/2$, an ℓ-step non-backtracking random walk is a convex combination of $(g-1)/2$-step non-backtracking random walks, which represent the last $(g-1)/2$ steps, whereas a $(g-1)/2$-step non-backtracking random walk is uniformly distributed on a set of $(d-1)^{(g-1)/2}$ vertices. Hence, an ℓ-step non-backtracking random walk returns to the start vertex with probability at most $1/(d-1)^{(g-1)/2}$, let alone doing so without revisiting any other vertex. ■

A version of Theorem 3.2 with an explicit construction. Turning back to the existence of robustly self-ordered graphs that do not have local self-ordering procedures, we point out that Theorem 3.2 does not provide an explicit construction of such graphs. An explicit construction, which carries a weaker query complexity lower bound for such procedures, is presented next.

Proposition 3.4 (explicit robustly self-ordered graphs that have no local self-ordering procedures): *For some constant $c > 0$, there exist explicit n-vertex bounded-degree graphs that are $\Omega(1)$-robustly self-ordered, but do not have local self-ordering procedures of query complexity $O(n^c)$. Furthermore, these graphs are explicit in a strong sense: given a vertex in the n-vertex graph, one can find its neighbors in* $\mathrm{poly}(\log n)$*-time.*

Proof Sketch: Intuitively, the construction of $\Omega(1)$-robustly self-ordered graphs presented in [4, Sec. 3] yields regular graphs that have logarithmic girth. Hence, by Proposition 3.3, the $\mathtt{cc}$-value of these n-vertex graphs is upper-bounded by $n^{-\Omega(1)}$, and Lemma 3.2.2 implies that a local self-ordering procedure for them requires $n^{\Omega(1)}$ queries.

The problem is that the construction presented in [4, Sec. 3] proceeds in two steps. The first step, taken in [4, Cor. 3.4], yields a 4-regular n-vertex directed (edge-colored) graph G, which is based on expanding generators for $\mathrm{SL}_2(p)$, that has logarithmic girth (when ignoring the direction of its edges). This directed graph is robustly self-ordered in the directed and colored sense (per [4, Def. 3.1]). In the second step, these directed and colored edges are replaced by constant-sized gadgets yielding a graph $\widehat{G}$ that is robustly self-ordered in the ordinary sense (see [4, Sec. 2&3.2]). However, these gadgets (specifically those used in [4, Sec. 2]) may either contain cycles or vertices of degree 1, and so either the girth of the graph or its regularity is not maintained. Nevertheless, we can first establish a lower bound on the query complexity of locally self-ordering G, and then obtain the same result for $\widehat{G}$ by a reduction.

Specifically, the argument skecthed above does hold for the n-vertex (directed) graph G; that is, letting $\overline{G}$ denote the undirected underlying graph of G, we note that $\overline{G}$ has logarithmic girth, and so $\mathtt{cc}(\overline{G}) = n^{-\Omega(1)}$ (by Proposition 3.3). By Lemma 3.2.2 a local self-ordering procedure for $\overline{G}$ requires $n^{\Omega(1)}$ queries, and the same holds for G.

Next, we reduce locally self-ordering G to locally self-ordering $\widehat{G}$, and doing so establish the same lower bound for $\widehat{G}$. Once this is done, the claim of the proposition is established for $\widehat{G}$. The reduction itself is straightforward: given oracle access to G, we can emulate (at constant overhead) oracle access to $\widehat{G}$, whereas locating a vertex (of G) in $\widehat{G}$ yields its location in G. ■

Can Theorem 3.2 be strengthened? Recall that Theorem 3.2 establishes the existence of $O(1)$-regular n-vertex graphs G that are $\Omega(1)$-robustly self-ordered such that any local self-ordering procedure for G requires $\Omega(\sqrt{n})$ queries. We ask whether this lower bound can be increased.

Open Problem 3.5 (what is the query complexity of local self-ordering for the worst possible robustly self-ordered graph): *Specifically,*

1. *Do there exist n-vertex bounded-degree graphs that are $\Omega(1)$-robustly self-ordered, but do not have local self-ordering procedures of query complexity $o(n)$.*
2. *Does every n-vertex bounded-degree graph that is $\Omega(1)$-robustly self-ordered have a local self-ordering procedure of query complexity $\widetilde{O}(\sqrt{n})$.*

We note that, in general, there exist n-vertex (asymmetric) graphs that require $\Omega(n)$ queries for local self-ordering: Consider an $(n-3)$-cycle that is augmented by an isolated vertex that is connected to one vertex and an 2-vertex path that is connected to its adjacent vertex (see Figure 6).

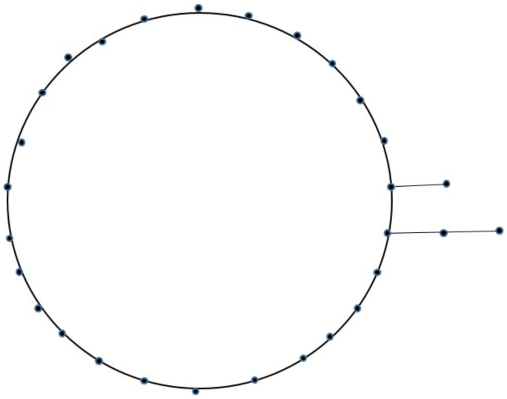

Figure 6: An augmented cycle requiring a linear number of queries for localization.

We also note that the lower bound asserted in Theorem 3.2 cannot be increased using the proof strategy used in our proof (i.e., using Lemma 3.2.2). Furthermore, it seems that such an improvement may not hold for random regular graphs, which (as shown in [4, Thm. 6.1]) are $\Omega(1)$-robustly self-ordered with probability $1-o(1)$.

Conjecture 3.6 (random regular graphs have $\widetilde{O}(\sqrt{n})$-query self-ordering procedures): *For every $d \geq 3$, with probability $1-o(1)$, a random d-regular n-vertex graph has a local self-ordering procedure of query complexity $\widetilde{O}(\sqrt{n})$.*

We believe that Conjecture 3.6 can be proved using the following observation. Let $t = O(\sqrt{\log n})$ and consider an arbitrary vertex v in such a random graph and t vertices, $r_1, ..., r_t$, that are at distance exactly $\lceil \log_{d-1} t \rceil$ from v. For each of these r_i's, consider the set S_i of vertices that are at distance exactly $\lceil \log_{d-1} n \rceil$ from r_i. Then, it seems that the sizes of the pairwise intersections between the S_i's uniquely identify v. Intuitively, this is the case since these sizes are sufficiently random, and so we expect to see different $\binom{t}{2}$-long sequences for the different n vertices (even under permutations of the t vertices).

Acknowledgements

I am grateful to Avi Wigderson for collaboration in early stages of this research.

References

[1] N. Alon, I. Benjamini, E. Lubetzky, and S. Sodin. Non-Backtracking Random Walks Mix Faster. *Communications in Contemporary Mathematics*, Vol. 9 (4), pages 585–603, 2007.

[2] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[3] O. Goldreich and L. Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, Vol. 23 (1), pages 23–57, August 2003.

[4] O. Goldreich and A. Wigderson. Robustly Self-Ordered Graphs: Constructions and Applications to Property Testing. *ECCC*, TR20-149, 2020.

[5] O. Goldreich and A. Wigderson. Non-adaptive vs Adaptive Queries in the Dense Graph Testing Model. *ECCC*, TR20-160, 2020.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il