

# Deterministic Even-Cycle Detection in Broadcast CONGEST \*

Pierre Fraigniaud<sup>1</sup>, Maël Luce<sup>1</sup>, Frédéric Magniez<sup>1</sup>, and Ioan Todinca<sup>2</sup>

<sup>1</sup> Université Paris Cité, CNRS, IRIF, Paris, France

<sup>2</sup> Université d'Orléans, INSA-Centre Val de Loire, LIFO, Orléans, France

## Abstract

We show that, for every  $k \geq 2$ ,  $C_{2k}$ -freeness can be decided in  $O(n^{1-1/k})$  rounds in the Broadcast CONGEST model, by a *deterministic* algorithm. This (deterministic) round-complexity is optimal for  $k = 2$  up to logarithmic factors thanks to the lower bound for  $C_4$ -freeness by Drucker et al. [PODC 2014], which holds even for *randomized* algorithms. Moreover it matches the round-complexity of the best known *randomized* algorithms by Censor-Hillel et al. [DISC 2020] for  $k \in \{3, 4, 5\}$ , and by Fraigniaud et al. [PODC 2024] for  $k \geq 6$ . Our algorithm uses parallel BFS-explorations with deterministic selections of the set of paths that are forwarded at each round, in a way similar to what was done for the detection of odd-length cycles, by Korhonen and Rybicki [OPODIS 2017]. However, the key element in the design and analysis of our algorithm is a new combinatorial result bounding the “local density” of graphs without  $2k$ -cycles, which we believe is interesting on its own.

## 1 Introduction

In the context of distributed computing in networks, deciding  $H$ -freeness for a given (connected) graph  $H$  has attracted a lot of attention in the standard CONGEST model (see, e.g., the survey [Cen21]). Indeed, this problem is inherently local, and thus the main concern is measuring the amount of information that must be exchanged between the nodes for solving it. Recall that the CONGEST model [Pel00] assumes  $n$  processing nodes connected as an  $n$ -node graph  $G = (V, E)$ . Computation proceeds as a sequence of rounds. During each round, every node can send an  $O(\log n)$ -bit message to each of its neighbors, receive the messages sent by its neighbors, and perform some individual computation. In the *broadcast* version of the model, which is the one used in this paper, it is required that, at each round, each node sends the same message to all its neighbors. An algorithm decides  $H$ -freeness if, for every graph  $G$ , the following holds: If  $G$  contains a subgraph isomorphic to  $H$  then at least one node *rejects*, otherwise all nodes *accept*.

For every  $k \geq 3$ , let  $C_k$  denote the  $k$ -node cycle. It is known that, for every  $k \geq 2$ , there exists a deterministic algorithm deciding  $C_{2k+1}$ -freeness in  $O(n)$  rounds [KR17], which is optimal up to logarithmic factors. It is however possible to decide the presence of even-size cycles in a sub-linear number of rounds. In particular, there exists a deterministic algorithm deciding  $C_4$ -freeness in  $O(\sqrt{n})$  rounds [DKO14], which is optimal up to logarithmic factors, even for randomized algorithms. For every  $k > 2$ , there exist randomized algorithms deciding  $C_{2k}$ -freeness in  $O(n^{1-1/k})$  rounds [CFG<sup>+</sup>20, FLMT24]. The algorithms in [CFG<sup>+</sup>20], based on the “local threshold” technique, apply to  $2 \leq k \leq 5$ , whereas the algorithms in [FLMT24], based on the “global threshold” technique, apply to all  $k \geq 2$ .

All aforementioned randomized algorithms are 1-sided, i.e., if  $G$  contains a  $2k$ -cycle, then the probability that at least one node rejects is at least  $2/3$ , but if  $G$  does not contain a  $2k$ -cycle, then the probability that all nodes accept is 1. Of course, the error probability of these algorithms can be made as small as

---

\*Research supported in part by the European QuantERA project QOPT (ERA-NET Cofund 2022-25), the French ANR projects DUCAT (ANR-20-CE48-0006), the French PEPR integrated project EPiQ (ANR-22-PETQ-0007), and the QuantEdu-France project (22-CMAS-0001).

desired by mere repetition. Yet, it may still be the case that  $G$  contains a  $2k$ -cycle that is not detected, even if this occurs with arbitrarily small probability. This raises the question of whether  $C_{2k}$ -freeness can be decided by a *deterministic* algorithm (which would thus provide absolute success) without increasing the round complexity. This is the case for all odd cycles of length  $\geq 5$ , and for 4-cycles [DKO14, KR17]. We show that this holds for all even cycles as well, by establishing the following result.

**Theorem 1.** *For every  $k \geq 2$ , there is a deterministic algorithm solving  $C_{2k}$ -freeness in  $O(n^{1-1/k})$  rounds in the Broadcast CONGEST model.*

Our deterministic algorithm solving  $C_{2k}$ -freeness in  $O(n^{1-1/k})$  rounds is generic, parameterized by  $k$ . For  $k = 2$ , i.e., for  $C_4$ -freeness, our algorithm coincides with the one in [DKO14]. In fact, our algorithm can be viewed as a generalisation of the latter. As for the algorithms in [CFG<sup>+</sup>20, FLMT24], they distinguish the case of *light* cycles, i.e.,  $2k$ -cycles containing solely nodes with degree smaller than  $n^{1/k}$ , from the case of *heavy* cycles, i.e.,  $2k$ -cycles containing at least one node with degree at least  $n^{1/k}$ . Light cycles can be easily detected deterministically by flooding, in  $O(n^{1-1/k})$  rounds [CFG<sup>+</sup>20, FLMT24], and the main issue is to show that heavy cycles can also be detected deterministically in  $O(n^{1-1/k})$  rounds. We show that this is indeed possible.

Our algorithm (for detecting heavy cycles) relies on combining the Representative Lemma by Monien [Mon85], with a new “Density Theorem”. In a nutshell, the Representative Lemma can be used to show that, as already observed in [FO19, KR17], for each source node  $v$ , it is not necessary for intermediate nodes to forward all prefixes of paths with endpoint  $v$  for eventually finding one path forming a cycle containing  $v$ , but forwarding a *constant* number of prefixes suffices. Our density theorem is used to show that, if a node  $v$  has to forward  $\omega(n^{1-1/k})$  prefixes of paths in total (i.e., corresponding to  $\omega(n^{1-1/k})$  source nodes  $v$ ), then there must exist a  $2k$ -cycle in the graph  $G$ . More precisely, our theorem states the following.

**Theorem 2.** *Let  $G = (V, E)$  be an  $n$ -node graph. For every  $v \in V$ , and every integer  $\ell \geq 1$ , let  $R_\ell(v) \subseteq V$  be the set of nodes that are reachable from  $v$  by a simple path of length exactly  $\ell$ . For every integer  $k \geq 2$ , if there exist  $v \in V$  and  $\ell \in \{1, \dots, k-1\}$  such that*

$$\sum_{u \in R_\ell(v)} \deg(u) > 6 \cdot (2k)^\ell \cdot n,$$

*then there is a  $2k$ -cycle in  $G$ .*

Combining the Representative Lemma by Monien [Mon85] with our density theorem, our algorithm for heavy  $2k$ -cycles boils down to flooding the network for  $k$  steps with path-prefixes originated at all heavy nodes  $v$ , under the following simple condition: at every step  $\ell \in \{1, \dots, k\}$ , if a node  $u$  has to forward prefixes coming from more than  $6 \cdot (2k)^\ell \cdot n^{1-1/k}$  heavy nodes then  $u$  rejects.

If flooding is completed without any rejection then the Representative Lemma guarantees that a  $2k$ -cycle will be detected if any. Instead, if flooding aborts at some rejecting nodes, then the density theorem guarantees that this rejection is legitimate as there must exist a  $2k$ -cycle.

## 2 Model and Preliminary Results

This section recalls the standard (Broadcast) CONGEST model, and takes benefit of the 4-cycle detection algorithm in [DKO14] for introducing the reader with some of the techniques that will be reused throughout the paper. This is particularly the case of the Representative Lemma by Monien [Mon85], which is the basis for implementing a filtering technique enabling to bound the congestion of cycle-detection algorithms.

### 2.1 The Broadcast CONGEST Model

The CONGEST model [Pel00] assumes  $n \geq 1$  processing nodes connected by a network modeled as an arbitrary  $n$ -node graph  $G = (V, E)$ . (All graphs are supposed to be connected and simple, i.e., no multiple

edges nor self-loops.) Each node  $v$  in  $G$  has an identifier  $\text{id}(v)$  taken from a polynomial range of identifiers, and thus encoded on  $O(\log n)$  bits. Each identifier is unique in the network. Computation proceeds as a sequence of synchronous rounds. All nodes start synchronously, at round 1. At each round, every node  $v$  can (1) send an  $O(\log n)$ -bit message to each of its neighbors in  $G$ , i.e., to each node  $u \in N(v)$ , (2) receive the messages sent by its neighbors, and (3) perform some individual computation. No limit is placed on the amount of computation that each node performs at each round. Initially, every node  $v$  knows its identifier  $\text{id}(v)$  and the size  $n$  of the graph it belongs to. No other information about the graph is provided to the nodes. All nodes perform the same algorithm, but the behavior of the nodes may vary along the course of execution of that algorithm, depending on the information acquired by them in each round (including their IDs).

The Broadcast CONGEST model is a restricted variant of the CONGEST model which requires each node to send the *same*  $O(\log n)$  bit message to all its neighbors, at every round. (Of course, the messages sent by a same node  $v$  at two different rounds may be different.)

## 2.2 Deciding $C_4$ -Freeness

The optimal (deterministic) algorithm for detecting 4-cycles in [DKO14] can be artificially rewritten as Algorithm 1. Let us say that a node  $v$  is *light* if its degree satisfies  $\deg(v) \leq \sqrt{2n}$ , and is *heavy* otherwise.

Phase 1 in Algorithm 1 is aiming at finding *light* 4-cycles, i.e., 4-cycles containing only light nodes. The for-loop of Instruction 5 consumes at most  $\sqrt{2n}$  rounds, as it involves light nodes only, i.e., the light node  $v$  has at most  $\sqrt{2n}$  neighbors, and thus at most  $\sqrt{2n}$  light neighbors. If a light 4-cycle is detected at Instruction 8, then  $v$  rejects appropriately.

Phase 2 is aiming at finding *heavy* 4-cycles, that is, 4-cycles containing at least one heavy node. The for-loop of Instruction 17 also consumes at most  $\sqrt{2n}$  rounds, merely because it is performed only if  $|\text{heavy}(v)| \leq \sqrt{2n}$ . The main observation is that it is legal for a node  $v$  to reject if  $|\text{heavy}(v)| > \sqrt{2n}$ . This is due to the following simple fact.

**Lemma 1.** *For every  $n$ -node graph  $G = (V, E)$ , if there exists  $v \in V$  such that the inequality  $\sum_{u \in N(v)} \deg(u) > 2n$  holds, then  $G$  contains a 4-cycle.*

*Proof.* **TOPROVE 0** □

Thanks to Lemma 1, since  $|\text{heavy}(v)| > \sqrt{2n}$  implies that  $\sum_{u \in N(v)} \deg(u) > 2n$ , we get that  $G$  contains a 4-cycle, and thus it is indeed legal for  $v$  to reject at Instruction 15.

Our generic algorithm for detecting  $2k$ -cycles for every  $k \geq 2$  follows the general idea of Algorithm 1 in the sense that:

1. it is also split into two phases, one for light cycles (i.e., containing only nodes with degree smaller than  $n^{1/k}$ ), and one for heavy cycles (i.e., containing at least one node with degree at least  $n^{1/k}$ ), and
2. it also utilizes a threshold as in Instruction 14, which is tuned depending on  $k$ .

In both phases, paths are broadcast among the nodes in the network. That is, every node  $v$  participating in the broadcast sends its identifiers to its neighbors, which concatenate their identifiers, and forward the resulting 1-edge path to their neighbors. After  $r$  rounds of such a process, every node  $u$  receives a set of paths, each of the form  $(\text{id}(w_0), \dots, \text{id}(w_{r-1}))$ , concatenates its identifier to each of these paths, and forwards the resulting set of paths, each of the form  $(\text{id}(w_0), \dots, \text{id}(w_{r-1}), \text{id}(u))$  to all its neighbors. In itself, such a process would however create huge congestion. Indeed, the number of paths circulating in the network would grow exponentially. Nevertheless, reducing drastically the number of paths can be achieved thanks to a simple application of the Representative Lemma by Monien [Mon85], as it was done in, e.g., [FO19, KR17], which is explained a bit further in the text. Before that, let us quickly cover the study of so-called *light* cycles, for which broadcast works without filtering.

---

**Algorithm 1** Algorithm executed by every node  $v \in V$  for deciding  $C_4$ -freeness in  $G = (V, E)$

---

```

1: send  $(\text{id}(v), \deg(v))$  to all neighbors
2: Phase 1: Looking for light 4-cycles
3: if  $\deg(v) \leq \sqrt{2n}$  then
4:    $\text{light}(v) \leftarrow \{u \in N(v) \mid \deg(u) \leq \sqrt{2n}\}$ 
5:   for  $u \in \text{light}(v)$  do
6:     send  $\text{id}(u)$  to all neighbors;
7:   end for
8:   if  $v$  has received  $\text{id}(w) \notin \{\text{id}(v), \text{id}(u), \text{id}(u')\}$  from  $u \neq u'$  both in  $\text{light}(v)$  then
9:     output(reject), and terminate
10:  end if
11: end if
12: Phase 2: Looking for heavy 4-cycles
13:  $\text{heavy}(v) \leftarrow \{u \in N(v) \mid \deg(u) > \sqrt{2n}\}$ 
14: if  $|\text{heavy}(v)| > \sqrt{2n}$  then
15:   output(reject)
16: else
17:   for  $u \in \text{heavy}(v)$  do
18:     send  $\text{id}(u)$  to all neighbors;
19:   end for
20:   if  $v$  has received  $\text{id}(w) \notin \{\text{id}(v), \text{id}(u), \text{id}(u')\}$  from  $u \neq u'$  both in  $N(v)$  then
21:     output(reject)
22:   else
23:     output(accept)
24:   end if
25: end if

```

---

## 2.3 Detection of Light $2k$ -Cycles

For every  $k \geq 2$ , the detection of *light*  $2k$ -cycles, i.e. of  $2k$ -cycles containing only nodes of degree at most  $n^{1/k}$ , can be done in a straightforward manner in  $O(n^{1-1/k})$  rounds. It is indeed sufficient to consider the subgraph  $G_{\text{light}}$  of the input graph  $G$  induced by the light nodes of  $G$ . The detection proceeds by looking for all  $2k$ -cycles  $G_{\text{light}}$  passing through  $v$ , for all nodes  $v \in V(G_{\text{light}})$  in parallel.

Specifically, the algorithm proceeds by flooding, during  $k$  phases. At the first phase (which lasts one round), every  $v \in V(G_{\text{light}})$  forms the path  $P = (v)$  consisting of the single node  $v$ , and sends it to all its neighbors in  $G_{\text{light}}$ . At every phase  $p \geq 2$ , for every node  $v \in V(G_{\text{light}})$ , and for every simple path  $P = (u_1, \dots, u_{p-1})$  received by  $v$  during phase  $p-1$ , if  $v \notin \{u_1, \dots, u_{p-1}\}$ , then  $v$  appends its identifier to  $P$  for forming the path  $P' = (u_1, \dots, u_{p-1}, v)$ , which is forwarded to all of  $v$ 's neighbors in  $G_{\text{light}}$ .

After  $k$  phases, if a node  $v$  has received a simple path  $P = (u_1, \dots, u_k)$  from a neighbor  $u_k$ , and a simple path  $P' = (u'_1, \dots, u'_k)$  from a neighbor  $u'_k$ , with  $u_1 = u'_1$ ,  $P \cap P' = \{u_1\}$ , and  $v \notin P \cup P'$ , then  $v$  rejects. Otherwise  $v$  accepts. We show that this simple flooding algorithm detects light  $2k$ -cycles in  $O(n^{1-1/k})$  rounds.

**Lemma 2.** *For every  $k \geq 2$ , there is a deterministic algorithm running in  $O(n^{1-1/k})$  rounds in  $n$ -node graphs under the Broadcast CONGEST model such that, for every graph  $G$ , if  $G$  contains a light  $2k$ -cycle (i.e., a cycle containing only nodes of degree smaller than  $n^{1/k}$ ) then at least one node rejects, otherwise all nodes accept.*

*Proof.* **TOPROVE 1** □

The main difficulty is detecting heavy cycles, that is, cycles containing at least one node of degree at least  $n^{1/k}$ . Among other techniques, one needs *filtered flooding*, explained in the next section.

## 2.4 Filtered Flooding

### 2.4.1 Representative Lemma

Monien [Mon85] defines a *representative* of a family of subsets of ground set  $[n] = \{1, \dots, n\}$  as follows.

**Definition 1.** For every integer  $n \geq 1$ , every family  $\mathcal{A} \subseteq 2^{[n]}$  of subsets of  $[n]$ , and every  $q \in [n]$ , a family of sets  $\mathcal{B} \subseteq \mathcal{A}$  is a  $q$ -representative of  $\mathcal{A}$  if, for every set  $X \subseteq [n]$  of size  $|X| \leq q$ , the following holds:

$$\exists A \in \mathcal{A} \text{ such that } A \cap X = \emptyset \iff \exists B \in \mathcal{B} \text{ such that } B \cap X = \emptyset.$$

Note that it follows directly from the definition that the  $q$ -representativity property is transitive, which is important as our algorithm for  $C_{2k}$ -freeness will perform several nested iterations of computing a  $q$ -representative set.

**Fact 1.** For every  $n \geq 1$ ,  $\mathcal{C} \subseteq \mathcal{B} \subseteq \mathcal{A} \subseteq 2^{[n]}$ , and  $q \in [n]$ , if  $\mathcal{B}$  is  $q$ -representative for  $\mathcal{A}$ , and  $\mathcal{C}$  is  $q$ -representative for  $\mathcal{B}$ , then  $\mathcal{C}$  is  $q$ -representative for  $\mathcal{A}$ .

The following lemma says that if the sets in the family  $\mathcal{A}$  have bounded size  $p$ , then there exists a  $q$ -representative of  $\mathcal{A}$  with bounded size.

**Lemma 3** (Monien [Mon85]). For every integer  $n \geq 1$ , every  $(p, q) \in [n] \times [n]$  such that  $p + q \leq n$ , and every family  $\mathcal{A} \subseteq 2^{[n]}$  of subsets of  $[n]$ , if  $|A| \leq p$  for all  $A \in \mathcal{A}$ , then there exists a  $q$ -representative family  $\mathcal{B} \subseteq \mathcal{A}$  of  $\mathcal{A}$  such that

$$|\mathcal{B}| \leq \binom{p+q}{p}.$$

To get a flavor of this lemma, observe that if  $p + q = n$  and  $\mathcal{A} = \{A \subseteq [n], |A| = p\}$ , then the bound of Lemma 3 is tight. Indeed,  $\mathcal{A}$  is then the set of all subsets of  $[n]$  of size  $p$ , meaning that  $|\mathcal{A}| = \binom{n}{p}$ . Now suppose that  $\mathcal{B} \subsetneq \mathcal{A}$ , then any  $A \in \mathcal{A} \setminus \mathcal{B}$  is such that  $A \cap ([n] \setminus A) = \emptyset$  but there is no  $B \in \mathcal{B}$  that does not intersect  $[n] \setminus A$ . The only  $q$ -representative family of  $\mathcal{A}$  is itself.

### 2.4.2 Application to Cycle Detection

The Representative Lemma (Lemma 3) is particularly useful in the context of  $2k$ -cycle detection for limiting congestion. Indeed, let us assume that one is questioning whether there is a  $2k$ -cycle in the  $n$ -node graph  $G = (V, E)$  containing a given node  $v \in V$ . One way to proceed is to let  $v$  broadcast its identifier for  $k$  rounds. That is, at step  $p = 0$ ,  $v$  sends  $\text{id}(v)$  to all its neighbors. Then, at step  $p = 1$ , every neighbor  $u \in N(v)$  appends its identifier to encode the 1-edge path  $(v, u)$ , and forwards this path to all its neighbors. More generally, if the flooding process is not filtered, then, at step  $p \in \{1, \dots, k-1\}$ , a node  $u$  receiving a simple path  $P = (v_0, \dots, v_{p-1})$  with  $v_0 = v$  appends  $u$  to  $P$  whenever  $u \notin \{v_0, \dots, v_{p-1}\}$ , and forwards the resulting augmented path to all its neighbors. At the end of step  $k-1$ , if a node  $u$  has received two simple paths  $P = (v_0, \dots, v_{k-1})$  and  $P' = (v'_0, \dots, v'_{k-1})$  with  $v_0 = v'_0 = v$ ,  $u \notin P \cup P'$ , and  $P \cap P' = \{v\}$ , then  $u$  has detected a  $2k$ -cycle containing  $v$ .

The absence of filtering in the above process may result in an exponential increase of the number of paths to be forwarded by an intermediate node  $u$ . This can be avoided using Lemma 3 by the following filtering process. Let us assume that, at step  $p \in \{1, \dots, k-1\}$ , a node  $u$  receives a collection  $\mathcal{A}$  of  $p$ -node simple paths, all with endpoint  $v$ . Each path can be viewed as a set of  $p$  nodes, i.e., a subset of the  $n$ -node set  $V$  with cardinality  $p$ . Let  $q = 2k - p$ . Lemma 3 says that there exists  $\mathcal{B} \subseteq \mathcal{A}$  of cardinality  $|\mathcal{B}| \leq \binom{2k}{p}$  such that, for every simple path  $X = (u_0, \dots, u_{2k-p-1})$  from  $u_0 = u$  to  $u_{2k-p-1} \in N(v)$ , if there exists a path  $A \in \mathcal{A}$  that does not intersect with  $X$ , i.e., such that  $A \cup X$  is a  $2k$ -cycle containing  $v$ , then there exists a path  $B \in \mathcal{B}$  that does not intersect with  $X$ , i.e., such that  $B \cup X$  is also a  $2k$ -cycle containing  $v$ . The filtering process consists for node  $u$  to forward the family  $\mathcal{B}$ , after concatenating itself to every path in it, instead of  $\mathcal{A}$ . By the filtering technique, we have the following.

**Fact 2.** *Each intermediate node  $u$  forwards at most  $\binom{2k}{p}$  paths of size  $p + 1$  and of endpoint  $v$  at each round  $p \in \{0, \dots, k - 1\}$ .*

Hence, the number of paths forwarded by a node  $u$  is constant for a fixed  $k$ . As a consequence, we get the following.

**Fact 3.** *For every  $k \geq 2$ , every  $n$ -node graph  $G = (V, E)$ , and every  $v \in V$ , checking whether there is a  $2k$ -path in  $G$  containing  $v$  takes at most  $\sum_{p=0}^{k-1} (p+1) \binom{2k}{p} < k2^{2k}$  rounds in the Broadcast **CONGEST** model.*

### 3 Deterministic Even-Cycle Detection

This section is dedicated to the proof of Theorem 1 assuming that the density theorem holds (cf. Theorem 2). The density theorem will be established in the next section. We start here by describing our algorithm for deciding  $C_{2k}$ -freeness, and then we proceed to the proof of Theorem 1.

#### 3.1 Algorithm Description

Algorithm 2 solves the detection of  $2k$ -cycles. As Algorithm 1, it is split into two phases, one aiming at detecting light  $2k$ -cycles, and one aiming at detecting heavy  $2k$ -cycles, where a  $2k$ -cycle is light if it contains only nodes of degree smaller than  $n^{1/k}$ , and it is heavy otherwise. The second phase, that is, the detection of heavy cycles, uses the filtering techniques (see Instruction 29) based on Lemma 3, and detailed in Section 2.4.2. The detection of light cycles has already been described and analyzed in Section 2.3, and this section focuses on Phase 2, i.e., the detection of heavy cycles, starting at Instruction 18.

In Algorithm 2, every node  $v$  of a graph  $G = (V, E)$  maintains a collection of local variables. The set  $\text{heavy}(v)$  contains all heavy neighbors of  $v$  in  $G$ , thanks to Instruction 1. For every  $w \in V$ , the set  $\mathcal{Q}(w, v)$  contains a collection of simple paths with endpoints  $v$  and  $w$ . At the beginning of  $\ell$ -th iteration of the for-loop of Instruction 23,  $\mathcal{Q}(w, v)$  contains paths of length exactly  $\ell$ , which are eventually updated at the end of the  $\ell$ -th iteration (cf. Instruction 35) to paths of length  $\ell + 1$ . At each iteration, the set  $W(v)$  is the set of nodes  $w$  such that there is at least one path from  $w$  to  $v$  in  $\mathcal{Q}(w, v)$ , i.e.,  $\mathcal{Q}(w, v)$  is not empty.

The main point in Algorithm 2 is the test performed at Instruction 25, which stipulates that if  $W(v)$  is too big, i.e., if  $v$  is connected to too many (heavy) nodes  $w$  by a path of length  $\ell$  at iteration  $\ell$ , then  $v$  rejects. If  $v$  does not reject, then it carries on the flooding of path-prefixes, by applying filtering for preserving the fact that, for each  $w \in W(v)$ ,  $|\mathcal{P}(w, v)| \leq \binom{2k}{\ell+1}$  at every iteration (cf. Fact 2). At the end of each iteration  $\ell$  of the for-loop of Instruction 23, node  $v$  appends  $\text{id}(v)$  to each path received during that iteration (see Instruction 35). That is, node  $v$  appends  $\text{id}(v)$  to each path  $P \in \mathcal{P}(w, u)$  not containing  $v$ , for all neighbors  $u \in N(v)$ , and it resets  $\mathcal{Q}(w, v)$  accordingly.

Finally, if node  $v$  has received two paths  $P$  and  $P'$  of length  $k$ , both in  $\mathcal{Q}(w, v)$  for some  $w \in V$ , i.e., both with endpoints  $v$  and  $w$ , such that the concatenation of  $P$  and  $P'$  forms a  $2k$ -cycle, then  $v$  rejects.

#### 3.2 Proof of Theorem 1

In absence of the threshold condition at Instruction 25, Algorithm 2 merely consists of building longer and longer paths between  $v$  and some nodes  $w \in V$ , such that if there exists  $v$  and  $w$  for which there exists two paths  $P$  and  $P'$  of length  $k$  between  $v$  and  $w$  that are internally disjoint, that is if  $P \cup P'$  form a  $2k$ -cycle,  $v$  will detect that fact, and reject accordingly. As already discussed before, the filtering of Instruction 29 does not prevent the algorithm from finding such paths, if any. The main issue is that Algorithm 2 stops at iteration  $\ell$  whenever  $|W(v)| > 6 \cdot (2k)^\ell \cdot n^{1-1/k}$ . Let us show that stopping under this condition is fine, as it implies the existence of a  $2k$ -cycle.

Let us assume that there exists a node  $v$  such that, at iteration  $\ell \in \{1, \dots, k - 1\}$  of the for-loop of Instruction 23,  $|W(v)| > 6 \cdot (2k)^\ell \cdot n^{1-1/k}$ . At iteration  $\ell$ ,  $W(v)$  is the set of *heavy* nodes  $w$  such that there is a simple path of length exactly  $\ell$  starting at  $w$  and ending at  $v$ . Using the notation of Theorem 2, let

---

**Algorithm 2** Algorithm run by every node  $v \in V$  for deciding  $C_{2k}$ -freeness in  $G = (V, E)$

---

```

1: send  $(\text{id}(v), \deg(v))$  to all neighbors
2: Phase 1: Looking for light  $2k$ -cycles
3: if  $\deg(v) < n^{1/k}$  then
4:   send  $P = (\text{id}(v))$  to all neighbors
5:   for  $i = 1$  to  $k - 1$  do
6:     receive paths sent by neighbors
7:     let  $\mathcal{P}$  be the set of received paths (not containing  $v$ )
8:     for  $P \in \mathcal{P}$  do
9:        $P \leftarrow (P, v)$ 
10:    send  $P$  to neighbors
11:   end for
12: end for
13: receive paths sent by neighbors, and let  $\mathcal{P}$  be the received set of paths
14: if  $\exists P, P' \in \mathcal{P}$  of same origin  $u$  such that  $v \notin P \cup P'$  and  $P \cap P' = \{u\}$  then
15:   output(reject) and terminate
16: end if
17: end if
18: Phase 2: Looking for heavy  $2k$ -cycles
19:  $\text{heavy}(v) \leftarrow \{u \in N(v) \mid \deg(u) \geq n^{1/k}\}$ 
20: for  $w \in V$  do
21:   if  $w \in \text{heavy}(v)$  then  $\mathcal{Q}(w, v) \leftarrow \{(\text{id}(w), \text{id}(v))\}$  else  $\mathcal{Q}(w, v) \leftarrow \emptyset$ 
22: end for
23: for  $\ell = 1$  to  $k - 1$  do
24:    $W(v) \leftarrow \{w \in V \mid \mathcal{Q}(w, v) \neq \emptyset\}$ ;
25:   if  $|W(v)| > 6 \cdot (2k)^\ell \cdot n^{1-1/k}$  then
26:     output(reject) and terminate
27:   else
28:     for  $w \in W(v)$  do
29:        $\mathcal{P}(w, v) \leftarrow$  filtering applied to  $\mathcal{Q}(w, v)$ 
30:       for  $P \in \mathcal{P}(w, v)$  do
31:         send  $P$  to neighbors
32:       end for
33:     end for
34:     for  $w \in V$  do
35:        $\mathcal{Q}(w, v) \leftarrow \{(P, v) \mid P \in \mathcal{P}(w, u) \wedge u \in N(v) \wedge v \notin P\}$ 
36:     end for
37:   end if
38: end for
39: if  $\exists w \in V, \exists P, P' \in \mathcal{Q}(w, v) \mid P \cup P' = C_{2k}$  then
40:   output(reject)
41: else
42:   output(accept)
43: end if

```

---

$R_\ell(v) \subseteq V$  be the set of nodes that are reachable from  $v$  by a simple path of length exactly  $\ell$ . We have  $W(v) \subseteq R_\ell(v)$ . It follows that

$$\sum_{w \in R_\ell(v)} \deg(w) \geq \sum_{w \in W(v)} \deg(w) \geq |W(v)| \cdot n^{1/k} > 6 \cdot (2k)^\ell \cdot n^{1-1/k} \cdot n^{1/k} = 6 \cdot (2k)^\ell \cdot n. \quad (1)$$

By Theorem 2,  $G$  contains a  $2k$ -cycle, and thus node  $v$  rejects rightfully.

It remains to show that the round complexity of Algorithm 2 is  $O(n^{1-1/k})$ . Phase 1, dedicated to the search of light  $2k$ -cycles, takes this many rounds, as established in Lemma 2. Let us show that Phase 2, dedicated to the search of heavy  $2k$ -cycles, has the same complexity. By Fact 3, for every node  $v$  and every heavy node  $w$ , the final set  $\mathcal{Q}(w, v)$  at iteration  $\ell = k - 1$  at Instruction 35 is built after at most  $k \cdot 2^{2k}$  rounds. By the threshold condition of Instruction 25, the number of families  $\mathcal{P}(w, v)$  to be transmitted by  $v$  is at most  $6 \cdot (2k)^\ell \cdot n^{1-1/k}$ . So, in total, Phase 2 of Algorithm 2 performs in at most  $k \cdot 2^{2k} \cdot 6 \cdot (2k)^\ell \cdot n^{1-1/k}$  rounds, that is  $O(n^{1-1/k})$  rounds for a fixed  $k$ . This completes the proof of Theorem 1 (under the assumption that Theorem 2 holds).  $\square$

## 4 Proof of Density Theorem

### 4.1 General Construction

This section is dedicated to the proof of Theorem 2. Let  $G = (V, E)$  be an  $n$ -node graph. Let  $k \geq 2$ ,  $\ell \in \{1, \dots, k-1\}$ , and  $v \in V$ . Let  $R(v)$  be the set of nodes of  $G$  that are reachable from  $v$  by a simple path of length exactly  $\ell$ , and let us assume that

$$\sum_{w \in R(v)} \deg(w) > 6 \cdot (2k)^\ell \cdot n.$$

Our goal is to show that there is a  $2k$ -cycle in  $G$ . In the following, we fix

$$\tau = 6 \cdot (2k)^\ell. \quad (2)$$

Let  $F$  be the set of edges incident to at least one node in  $R(v)$ . Let  $F_{int} \subseteq F$  be the set of edges whose both endpoints are in  $R(v)$ , and let  $F_{ext} = F \setminus F_{int}$ .

**Lemma 4** (S. Burr [Bur82]). *Every  $m$ -edge graph contains a bipartite subgraph with at least  $m/2$  edges.*

Thanks to Lemma 4, there exists a bipartite subgraph of the graph  $G[F_{int}]$  induced by  $F_{int}$  with at least  $|F_{int}|/2$  edges. Let  $F'_{int} \subseteq F_{int}$  be the set of edges of this bipartite graph, hence  $|F'_{int}| \geq |F_{int}|/2$ .

Furthermore,  $F_{ext}$  also induces a bipartite subgraph as all of its edges have exactly one end in  $R(v)$ .

Let  $H$  be the bipartite graph defined as the subgraph of  $G$  induced by  $F'_{int}$  if  $|F'_{int}| \geq |F_{ext}|$ , or as the subgraph of  $G$  induced by  $F_{ext}$  otherwise. That is,

$$H = \begin{cases} G[F'_{int}] & \text{if } |F'_{int}| \geq |F_{ext}| \\ G[F_{ext}] & \text{otherwise.} \end{cases}$$

Let  $(X, Y)$  be a partition of the vertices of  $H$ , such that

$$X \subseteq R(v).$$

Note that some nodes in  $Y$  may also belong to  $R(v)$ .  $H$  satisfies the following.

**Fact 4.**  $H = (X, Y, E_H)$  is a bipartite subgraph of  $G$  with at least  $\frac{1}{6} \tau n$  edges.

*Proof.* **TOPROVE 2**  $\square$



To prove the existence of a  $2k$ -cycle in  $G$ , we will prove that there exist three simple paths  $P$ ,  $P'$ , and  $P''$  in  $G$  such that:

- $P$  is of length  $\lambda$  for some  $\lambda \in \{1, \dots, \ell\}$  connecting a node  $x \in X$  to some node  $u \in V$ .
- $P'$  is a path in  $H$  of length  $2k - 2\lambda - 1$  connecting  $x$  to a node  $y \in Y$ . Note that, since  $H$  is bipartite,  $P'$  alternates between nodes in  $X$  and nodes in  $Y$ .
- $P''$  is a path of length  $\lambda + 1$  connecting  $y$  to  $u$ .

Moreover, our construction will guarantee that  $P$ ,  $P'$  and  $P''$  are internally disjoint, in the sense that  $P \cap P' = \{x\}$ ,  $P \cap P'' = \{u\}$ , and  $P' \cap P'' = \{y\}$ . This is sufficient for establishing Theorem 2 as  $P \cup P' \cup P''$  is a  $2k$ -cycle in  $G$ . To exhibit these three paths, let us introduce some notations.

## 4.2 The Sets In and Out.

For every  $u \in V$ , and every  $i \in \{0, \dots, \ell\}$ , let us denote by  $\mathcal{Q}_i(u)$  the set of simple paths of length exactly  $i$  with one endpoint equal to  $u$ , and the other endpoint equal to some node in  $X$ .

For every  $u \in V$ , and  $i \in \{0, \dots, \ell\}$ , we define the two sets  $\text{in}_i(u)$  and  $\text{out}_i(u)$  as subsets of edges from  $H$ . Intuitively, the set  $\text{out}_i(u)$  can be viewed as a set of edges that  $u$  sends to its neighbors at round  $i$  in a (virtual) distributed protocol that broadcasts sets of edges of  $H$  throughout the network  $G$ , and  $\text{in}_i(u)$  can be viewed as a set of edges that  $u$  receives from its neighbors at round  $i$  of this protocol. Let  $E_H(u)$  denote the set of edges incident to  $u$  in  $H$ . For every  $u \in V$ , let

$$\text{out}_0(u) = \begin{cases} E_H(u) & \text{if } u \in X, \\ \emptyset & \text{otherwise.} \end{cases} \quad (3)$$

That is,  $\text{out}_0(u)$  can be viewed as the initialization of the aforementioned (virtual) broadcast protocol, i.e., initially, every node in  $X$  sends its incident edges in  $H$  to its neighbors in  $G$ . Now, let us define the sets  $\text{in}_i(u)$  and  $\text{out}_i(u)$  for every  $u \in V$  and  $i \geq 1$ , where, again,  $\text{in}_i(u)$  can be viewed as the set of edges received by  $u$  at round  $i$  (which were sent by  $u$ 's neighbors at round  $i - 1$ ), and  $\text{out}_i(u)$  can be viewed as the set of edges forwarded by  $u$  at round  $i$ . A key point is that  $u$  does not forward all the received edges, but a carefully chosen subset of these edges.

Formally, for every  $u \in V$  and every  $i \in \{1, \dots, \ell\}$ , let

$$\text{in}_i(u) = \bigcup_{\{w \in N(u) \mid \exists P \in \mathcal{Q}_{i-1}(w), (P, u) \in \mathcal{Q}_i(u)\}} \text{out}_{i-1}(w). \quad (4)$$

That is,  $\text{in}_i(u)$  merges all edges from sets  $\text{out}_{i-1}(w)$  for all neighbors  $w$  of  $u$  such that at least one path in  $\mathcal{Q}_{i-1}(w)$  can be extended into a path in  $\mathcal{Q}_i(u)$  (see Figure 1).

To define  $\text{out}_i(u)$ , we first define the sets  $\text{in}_i(w, u)$  and  $\text{out}_i(w, u)$ , the subsets of edges respectively in  $\text{in}_i(u)$  and  $\text{out}_i(u)$  incident to node  $w \in X \cup Y$  (see Figure 2). For every  $i \in \{1, \dots, \ell\}$ , for every node  $w$  of  $H$ , i.e., for every  $w \in X \cup Y$ , and every  $u \in V$ , let

$$\text{in}_i(w, u) = E_H(w) \cap \text{in}_i(u). \quad (5)$$

For every vertex  $u \in V$ , and every  $i \in \{1, \dots, \ell\}$ , we construct the edge-set  $\text{out}_i(u)$  by defining, for each  $y \in Y$ , a subset  $\text{out}_i(y, u)$  of  $\text{out}_i(u)$  containing edges of  $\text{out}_i(u)$  incident to  $y$ . First, for every  $y \in Y$  and  $u \in V$ , we set

$$\text{out}_0(y, u) = E_H(y) \cap \text{out}_0(u) = \begin{cases} \{y, u\} & \text{if } \{y, u\} \in E_H \\ \emptyset & \text{otherwise} \end{cases}$$

For every  $u \in V$  and  $i \in \{1, \dots, \ell\}$ , and for every  $y \in Y$ , we set

$$|\text{in}_i(y, u)| < (2k)^i \implies \text{out}_i(y, u) = \text{in}_i(y, u). \quad (6)$$

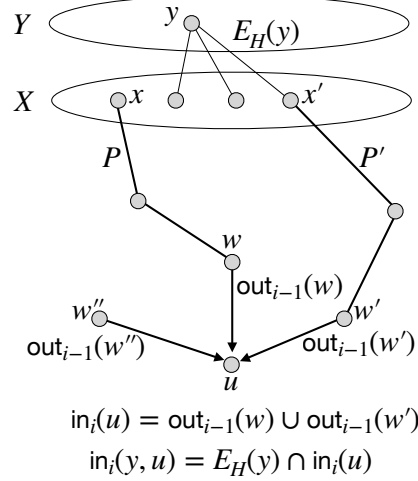


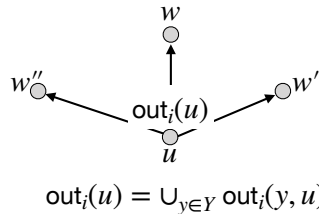
Figure 1: Construction of the set  $\text{in}_i(u)$  from the sets  $\text{out}_{i-1}(w)$ ,  $w \in N(u)$ . In the figure,  $\text{in}_i(u) = \text{out}_{i-1}(w) \cup \text{out}_{i-1}(w')$  because there is a simple path  $P$  (resp.,  $P'$ ) of length  $i - 1$  from  $w$  (resp.,  $w'$ ) to  $X$  that can be extended to a simple path of length  $i$  from  $u$  to  $X$ . For every  $y \in Y$ ,  $\text{in}_i(y, u) = E_H(y) \cap \text{in}_i(u)$ .

The definition of  $\text{out}_i(y, u)$  when  $|\text{in}_i(y, u)| \geq (2k)^i$  requires more care. Let  $H_i^{(u)}$  be the subgraph of  $H$  induced by all edges in  $\cup_{y \in Y} \text{in}_i(y, u)$  where one keeps only the large sets  $\text{in}_i(y, u)$ , that is,

$$H_i^{(u)} = G \left[ \bigcup_{\{y \in Y : |\text{in}_i(y, u)| \geq (2k)^i\}} \text{in}_i(y, u) \right]. \quad (7)$$

Note that, by construction, every edge  $e \in \text{in}_i(u) = \cup_{y \in Y} \text{in}_i(y, u)$  is either in  $H_i^{(u)}$ , or in the set  $\text{out}_i(y, u)$  where  $y \in Y$  is incident to  $e$ .

We are now going to update  $H_i^{(u)}$  iteratively, by repeating the following sequence of “peeling” operations as long as they can be applied, i.e., as long as vertices can be removed. This sequence of operations bears similarities with the computation of the  $k$ -core of a graph, but the vertices of the partitions  $X$  and  $Y$  of  $H$  are here treated separately.



$$\text{out}_i(y, u) = \begin{cases} \text{in}_i(y, u) & \text{if } |\text{in}_i(y, u)| < (2k)^i \\ E_{H_i^{(u)}}(y) & \text{if } |\text{in}_i(y, u)| \geq (2k)^i \text{ and } y \text{ is removed by peeling} \\ \emptyset & \text{otherwise} \end{cases}$$

Figure 2: Construction of the set  $\text{out}_i(u) = \cup_{y \in Y} \text{out}_i(y, u)$  from the sets  $\text{in}_i(y, u)$ ,  $y \in Y$ .

The peeling process applied to  $H_i^{(u)}$

Repeat until no nodes can be removed:

1. Remove from  $H_i^{(u)}$  all vertices  $x \in X$  of degree smaller than  $k$ , along with their incident edges in  $H_i^{(u)}$ , and update the degree of each vertex in  $H_i^{(u)}$  accordingly;
2. Remove from  $H_i^{(u)}$  all vertices  $y \in Y$  of degree smaller than  $k$ , along with their incident edges in  $H_i^{(u)}$ , and update the degree of each vertex in  $H_i^{(u)}$  accordingly;
3. For every  $y \in Y$ , we set

$$y \text{ removed at Instruction 2} \implies \text{out}_i(y, u) = E_{H_i^{(u)}}(y) \text{ (i.e. the edges removed with } y \text{)} ; \quad (8)$$

That is,  $\text{out}_i(y, u)$  is defined as the set of edges incident to  $y$  that were removed from  $H_i^{(u)}$  together with  $y$  at this iteration.

For any triple  $i \geq 1$ ,  $u \in V$ , and  $y \in Y$  satisfying  $|\text{in}_i(y, u)| \geq (2k)^i$ , if the value of  $\text{out}_i(y, u)$  has not been set in the above process, then it is set to  $\text{out}_i(y, u) = \emptyset$ . That is, for every  $y \in Y$ ,

$$\text{Eqs. (6) and (8) do not apply} \implies \text{out}_i(y, u) = \emptyset.$$

Finally, we set

$$\text{out}_i(u) = \bigcup_{y \in Y} \text{out}_i(y, u). \quad (9)$$

Note that, for every  $i \geq 1$ ,  $u \in V$ , and  $y \in Y$ , we have

$$\text{out}_i(y, u) = E_H(y) \cap \text{out}_i(u).$$

This equality is extended to define, for every triple  $i \geq 1$ ,  $u \in V$ , and  $x \in X$ ,

$$\text{out}_i(x, u) = E_H(x) \cap \text{out}_i(u).$$

### 4.3 Core Graphs.

The graph resulting from the above iterated process of edge- and vertex-removal from  $H_i^{(u)}$  is denoted by  $\text{Core}_i(u)$ . The core graphs play an important role in our proof. Indeed, we will show (see Lemma 6) that if  $\text{Core}_i(u)$  is non-empty for some  $i \geq 1$  and  $u \in V$ , then  $G$  contains a  $2k$ -cycle. The density theorem will then follow from the fact (established in Lemma 7) that there exists  $i$  and  $u$  such that  $\text{Core}_i(u)$  is non-empty.

Before proving Lemmas 6 and 7, let us establish a collection of statements for helping understanding the construction above. The fact below illustrates why one can view the sets  $\text{out}_i(u)$  and  $\text{in}_i(u)$  as produced by a (virtual) protocol broadcasting edges of  $H$  throughout the network  $G$ .

**Fact 5.** *For every simple path  $(u_0, \dots, u_\ell)$  in  $G$  with  $u_0 \in X$ , we have that, for every  $i \in \{1, \dots, \ell\}$ ,*

$$\text{out}_{i-1}(u_{i-1}) \subseteq \text{in}_i(u_i).$$

*Proof.* TOPROVE 3 □

**Lemma 5.** *For every  $i \in \{0, \dots, \ell\}$ ,  $u \in V$ , and  $e = \{x, y\} \in \text{in}_i(u)$  with  $x \in X$  and  $y \in Y$ , there is a simple path  $(u_0, \dots, u_i)$  in  $G$  such that  $u_0 = x$ ,  $u_i = u$ , and  $e \in \bigcap_{j=0}^{i-1} \text{out}_j(y, u_j)$ .*

To gain intuition about this statement, one can follow the idea of the virtual distributed protocol mentioned since the beginning of Section 4.2. This lemma states that any edge  $e = (x, y)$  that was received by  $u$  during the virtual protocol was first broadcast by  $x$  itself and forwarded along a path from  $x$  to  $u$ .

*Proof.* **TOPROVE 4** □

**Fact 6.** For all  $i \in \{1, \dots, \ell\}$  and  $u \in V$ , every node  $y \in Y \cap \text{Core}_i(u)$  satisfies  $|\text{in}_i(y, u)| \geq (2k)^i$ .

*Proof.* **TOPROVE 5** □

**Fact 7.** For all  $i \in \{1, \dots, \ell\}$  and  $u \in V$ , every node  $w \in \text{Core}_i(u)$  is of degree at least  $k$  in the graph  $\text{Core}_i(u)$ .

*Proof.* **TOPROVE 6** □

**Fact 8.** For every  $i \in \{1, \dots, \ell\}$ ,  $y \in Y$ , and  $u \in V$ , we have  $|\text{out}_i(y, u)| < (2k)^i$ .

*Proof.* **TOPROVE 7** □

**Fact 9.** For every  $i \in \{1, \dots, \ell\}$ ,  $u \in V$ , and  $x \in X$ , we have

$$|\text{in}_i(x, u)| \leq |\text{out}_i(x, u)| + \deg_{\text{Core}_i(u)}(x) + k - 1.$$

*Proof.* **TOPROVE 8** □

We are now ready to establish one of the two main arguments in the proof of our density theorem.

**Lemma 6.** If there exist  $i \in \{1, \dots, \ell\}$  and  $u \in V$  such that  $\text{Core}_i(u) \neq \emptyset$  then there is a  $2k$ -cycle in  $G$ .

*Proof.* **TOPROVE 9** □

**Lemma 7.** There exists  $i \in \{1, \dots, \ell - 1\}$  and  $u \in V$  such that  $\text{Core}_i(u) \neq \emptyset$ .

*Proof.* **TOPROVE 10** □

*Proof.* **TOPROVE 11** □

## 5 Conclusion

We have proved that, for every  $k \geq 2$ , there exists a *deterministic* distributed algorithm that decides whether the input graph  $G$  is  $C_{2k}$ -free in  $O(n^{1-1/k})$  rounds under the **CONGEST** model. This result is based on a new result in graph theory, which essentially states that when some form of “local density” exceeds a certain threshold (that depends on  $k$ ) in a graph, that graph must contain a  $2k$ -cycle.

We point out that our deterministic algorithm for  $C_{2k}$ -freeness can be used to solve the same problem in the Quantum **CONGEST** model in  $\tilde{O}(n^{1/2-1/2k})$  rounds, following the same approach as in [FLMT24]. Informally, the approach consists in three steps : (1) apply the “diameter reduction” technique of [EFF<sup>+</sup>22], which allows to reduce the problem to graphs of polylogarithmic diameter, (2) operate a “congestion reduction” on Algorithm 2 to make it work in a constant number of **CONGEST** rounds, at the cost of reducing the probability for cycle detection to  $\Theta(n^{-1+1/k})$ , and (3) eventually use an “amplification technique” based on quantum Grover search to obtain a constant probability of detecting a  $C_{2k}$ , if exists, in  $\tilde{O}(n^{1/2-1/2k})$  rounds. The same round complexity was already attained in [FLMT24], but the new algorithm is considerably simpler.

Note that the constant  $6 \cdot (2k)^\ell$  in our density theorem is not tight, at least for some values of  $k$  and  $\ell$ . For instance, for  $k = 2$ , and  $\ell = 1$ , our theorem states that if there exists  $v \in V$  such that  $\sum_{u \in N(v)} \deg(u) > 24 \cdot n$ , then there is a 4-cycle in  $G$  whereas, in fact, there is a 4-cycle in  $G$  already when  $\sum_{u \in N(v)} \deg(u) > 2 \cdot n$ . We let as an open problem the determination, for every  $k \geq 2$  and  $\ell \in \{1, \dots, k - 1\}$ , of the smallest value  $\tau = \tau(k, \ell)$  such that the existence of a node  $v$  for which  $\sum_{u \in R_\ell(v)} \deg(u) > \tau \cdot n$  implies the existence of a  $2k$ -cycle.

Arguably one of the main open problems in the field of distributed subgraph detection under the **CONGEST** model is whether  $O(n^{1-1/k})$  is the best round-complexity that can be achieved for  $2k$ -cycle detection, whether it be by deterministic or randomized algorithms, up to polylogarithmic factors. It is known to be the case for  $k = 2$ , i.e., the round-complexity of  $C_4$ -freeness is  $\tilde{\Omega}(\sqrt{n})$ , but it is open for  $k > 2$ . In other words, is it true that, for every  $k \geq 2$ ,  $C_{2k}$ -freeness cannot be decided under the **CONGEST** model in  $\tilde{o}(n^{1-1/k})$  rounds?

## References

- [Bur82] Stefan A Burr. Antidirected subtrees of directed graphs. *Canadian Mathematical Bulletin*, 25(1):119–120, 1982.
- [Cen21] Keren Censor-Hillel. Distributed subgraph finding: Progress and challenges. In *48th Int. Coll. on Automata, Languages, and Programming (ICALP)*, volume 198 of *LIPIcs*, pages 3:1–3:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [CFG<sup>+</sup>20] Keren Censor-Hillel, Orr Fischer, Tzlil Gonen, François Le Gall, Dean Leitersdorf, and Rotem Oshman. Fast distributed algorithms for girth, cycles and small subgraphs. In *34th International Symposium on Distributed Computing (DISC)*, volume 179 of *LIPIcs*, pages 33:1–33:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- [DKO14] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *33rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 367–376, 2014.
- [EFF<sup>+</sup>22] Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. Sublinear-time distributed algorithms for detecting small cliques and even cycles. *Distributed Computing*, 35(3):207–234, June 2022.
- [FLMT24] Pierre Fraigniaud, Maël Luce, Frédéric Magniez, and Ioan Todinca. Even-cycle detection in the randomized and quantum CONGEST model. In *43rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 209–219. ACM, 2024.
- [FO19] Pierre Fraigniaud and Dennis Olivetti. Distributed detection of cycles. *ACM Trans. Parallel Comput.*, 6(3):12:1–12:20, 2019.
- [KR17] Janne H. Korhonen and Joel Rybicki. Deterministic subgraph detection in broadcast CONGEST. In *21st International Conference on Principles of Distributed Systems (OPODIS)*, volume 95 of *LIPIcs*, pages 4:1–4:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
- [Mon85] Burkhard Monien. How to find long paths efficiently. In *North-Holland Mathematics Studies*, volume 109, pages 239–254. Elsevier, 1985.
- [Pel00] David Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, 2000.