Deterministic Subgraph Detection in Broadcast CONGEST*

Janne H. Korhonen[1] and Joel Rybicki[2]

1 **Aalto University, Finland**
`janne.h.korhonen@aalto.fi`
2 **University of Helsinki, Finland**
`joel.rybicki@helsinki.fi`

Abstract

We present simple deterministic algorithms for subgraph finding and enumeration in the broadcast CONGEST model of distributed computation:

- For any constant k, detecting k-paths and trees on k nodes can be done in $O(1)$ rounds.
- For any constant k, detecting k-cycles and pseudotrees on k nodes can be done in $O(n)$ rounds.
- On d-degenerate graphs, cliques and 4-cycles can be enumerated in $O(d + \log n)$ rounds, and 5-cycles in $O(d^2 + \log n)$ rounds.

In many cases, these bounds are tight up to logarithmic factors. Moreover, we show that the algorithms for d-degenerate graphs can be improved to $O(d/\log n)$ and $O(d^2/\log n)$, respectively, in the supported CONGEST model, which can be seen as an intermediate model between CONGEST and the congested clique.

1998 ACM Subject Classification C.2.4 Distributed Systems, G.2.2 Graph Theory

Keywords and phrases distributed computing, subgraph detection, CONGEST model, lower bounds

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2017.4

1 Introduction

Subgraph detection is a fundamental problem in algorithmics: given a fixed target graph H and an input graph G, the task is to decide whether G contains a subgraph isomorphic to H. This problem has been extensively studied in centralised algorithmics, and it is known that depending on the graph H, this problem can have very different complexities – for example, if H is a path, the problem can be solved in linear (in the number of nodes in G) time [1], but for cliques this is thought not to be the case [12].

In this work, we study this problem in the CONGEST model of distributed computation. So far, most work has either focused on (a) lower bounds for subgraph detection [13], (b) randomised upper bounds [21, 17], or (c) property testing of H-freeness [20, 8, 17]. By contrast, we focus on deterministic upper bounds, showing that simple techniques – indeed, often techniques well-known in non-distributed algorithmics – result in essentially optimal algorithms. Moreover, all our results work also in the weaker *broadcast* CONGEST model, where nodes send the same message to all neighbours in each communication round.

* This work was supported in part by the Academy of Finland, Grants 285721 and 1273253.

Results: subgraph detection on general graphs. First, we give simple constant-time detection algorithms for the case where H is either a path or a tree. These algorithms are essentially translations of known centralised *fixed-parameter* algorithms, based on *representative families* [23, 18, 24]. Specifically, we show that in the broadcast CONGEST model,

- k-paths can be detected in $O(k2^k)$ rounds, and
- any tree on k nodes can be detected in $O(k2^k)$ rounds[1].

Using these algorithms as building blocks, we then give linear-time detection algorithms for cycles and pseudotrees:

- k-cycles can be detected in $O(k2^k n)$ rounds, and
- any pseudotree (a graph with exactly one cycle) on k nodes can be detected in $O(k2^k n)$ rounds.

All algorithms can be implemented so that any node that detects the existence of a copy of H can also output full information about a single copy of H.

For odd $k \geq 5$, Drucker et al. [13] have proven a lower bound of $\Omega(n/\log n)$ rounds for detecting k-cycles, meaning that our cycle detection algorithm is optimal up to a logarithmic factor for a constant odd k. For 4-cycles, Drucker et al. [13] gave a lower bound of $\Omega(n^{1/2}/\log n)$ rounds, and a weaker lower bound for longer even cycles (see Section 2); we prove a simple extension of the Drucker et al. [13] lower bound by showing that for all even $k \geq 6$, detecting k-cycles requires $\Omega(n^{1/2}/\log n)$ rounds in the CONGEST model.

Results: subgraph enumeration on sparse graphs. Second, we study subgraph detection and enumeration on sparse input graphs. Specifically, we study a setting where the input graph has small *degeneracy*; in distributed and parallel computing, this setting has been studied in the context of e.g. symmetry breaking in the LOCAL model [2, 4] and enumerating triangles and 4-cliques of planar graphs in the PRAM model [10]. Denoting by d the degeneracy of the input graph, we show that in the broadcast CONGEST model,

- k-cliques can be enumerated in $O(d + \log n)$ rounds for any k,
- 3-cycles and 4-cycles can be enumerated in $O(d + \log n)$ rounds, and
- 5-cycles can be enumerated in $O(d^2 + \log n)$ rounds,

where enumeration means that every copy of the target subgraph H is output by some node in the network.

This is as far as we can push this framework; we show that already detecting k-cycles for $k \geq 6$ requires $\Omega(n^{1/2}/\log n)$ rounds on graphs of degeneracy 2. Moreover, a careful examination of the Drucker et al. [13] lower bound constructions show that detecting 4-cycles and 5-cycles requires $\Omega(d/\log n)$ rounds.

Finally, we discuss how the results on detecting subgraphs in sparse graphs can be translated into the *supported* CONGEST *model* proposed by Schmid and Suomela [29]. In this model, we can close many of the logarithmic gaps between the upper and lower bounds that remain in the CONGEST model.

---

[1] We note that the constant-round algorithms for path and tree detection were also independently discovered by Fraigniaud et al. [19]; however, we give more detailed analysis in terms of the factors dependent on k. In fact, very similar algorithms were independently discovered in *four* separate works almost concurrently, including this one [22] and the papers of Fraigniaud et al. [19], Even et al. [16] and Fischer et al. [17]. The last three appear as a joint paper in DISC 2017 [15]. While our work is otherwise independent of the others, our pseudotree detection algorithm is motivated by the pseudotree property testing results of Fraigniaud et al. [19].

2 Related work

Subgraph detection upper bounds. For deterministic subgraph detection in the CONGEST model, the only prior works we are aware of are the $O(n^{1/2})$ round algorithm for 4-cycle detection by Drucker et al. [13], and the independent discovery of the constant-round path and tree detection algorithms [19, 15]. In the *congested clique* model, deterministic subgraph detection algorithms were given by Dolev et al. [11] and Censor-Hillel et al. [9]; the latter is in particular noteworthy from our perspective, as it used *colour-coding* techniques from centralised fixed-parameter algorithmics to obtain fast cycle detection algorithms for cycles of arbitrary length.

Randomised subgraph detection and listing in the CONGEST model has been recently receiving attention from multiple authors. Izumi and Le Gall [21] gave an $\tilde{O}(n^{2/3})$ round algorithm for detecting triangles, and a $O(n^{3/4}\log n)$ round algorithm for enumerating triangles. Independent of our work, Fischer et al. [17] gave a colour-coding algorithm that can detect any constant-size tree on k nodes with constant probability in $O(k^k)$ rounds.

Subgraph detection lower bounds. As noted before, Drucker et al. [13] have studied lower bounds for cycle detection in the CONGEST model. Specifically, their lower bound for k-cycle detection is $\Omega(\mathrm{ex}(n, C_k)/n \log n)$ rounds, where $\mathrm{ex}(n, C_k)$ is the *Turán number* for cycles, that is, the maximum number of edges in a k-cycle-free graph with n nodes. This lower bound is $\Omega(n/\log n)$ for odd cycles, and $\Omega(n^{1/2}/\log n)$ for 4-cycles. However, for longer even cycles, this can give at most $\Omega(n^{1+2/k}/\log n)$ due to known bounds for Turán numbers, and matching bounds on Turán numbers are only known for $k = 6$ and $k = 8$; see e.g. Pikhurko [28] and references therein.

To our knowledge, no other subgraph detection lower bounds are known in the CONGEST model. In particular, proving lower bounds for triangle detection seems to be a particularly difficult challenge. However, for the broadcast congested clique model, Drucker et al. [13] give an $\Omega(n/e^{\sqrt{\log n}}\log n)$ round lower bound, which also applies to the broadcast CONGEST model. Moreover, for triangle enumeration lower bounds are known, also in the stronger congested clique model [21, 26].

Property testing for H-freeness. Property testing of H-freeness in the CONGEST model is another question that has received a lot of attention lately. In this setting, an algorithm has to correctly decide with probability at least 2/3 if the input graph is (a) H-free – that is, does not contain a subgraph isomorphic to H – or (b) ε-away from being H-free; in the intermediate case, the algorithm can perform arbitrarily. See e.g. Censor-Hillel et al. [8] for complete definitions.

Property testing algorithms for triangle-freeness were given by Censor-Hillel et al. [8]; for H-freeness for graphs on 4 nodes by Fraigniaud et al. [19] and Even et al. [16]; and for H-freeness for most graphs on 5 nodes by Fischer et al. [17] Moreover, property testing algorithms for tree and cycle freeness – using techniques of fixed-parameter algorithmics flavour – have been discovered recently [16, 17, 19, 15].

3 Preliminaries

Set notation. Let $\mathbb{N} = \{0, 1, \ldots\}$ be the set of non-negative integers. For integer $n \geq 1$, we use the notation $[n] = \{1, 2, \ldots, n\}$. For any set A, we use $2^A = \{B : B \subseteq A\}$ to denote the power set of A.

Graphs. A graph is a pair $G = (V, E)$, where V is the set of nodes and $E \subseteq 2^V$ is the set of edges. We use $n = |V|$ to denote the number of nodes in the graph. An *orientation* σ of graph G is a labelling of the edges that assigns a direction $\sigma(\{u, v\}) \in \{u \to v, u \leftarrow v\}$ to every edge $\{u, v\} \in E$.

The open neighbourhood of a node $v \in V$ is the set $N(v) = \{u \in V : \{u, v\} \in E\}$ and the closed neighbourhood is the set $N^+(v) = N(v) \cup \{v\}$. The degree of a node $v \in V$ is $\deg(v) = |N(v)|$. The neighbours of $v \in V$ with incoming and outgoing edges under an orientation σ are denoted by $N_{\text{in}}(v) = \{u \in N(v) : \sigma(\{u, v\}) = u \to v\}$ and $N_{\text{out}}(v) = N(v) \setminus N_{\text{in}}(v)$. The indegree of $v \in V$ under an orientation σ is $\text{indeg}(v) = |N_{\text{in}}(v)|$ and the outdegree is given by $\text{outdeg}(v) = |N_{\text{out}}(v)|$.

For a graph G, a subgraph $G' \subseteq G$ of G is a graph $G' = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E \cap 2^{V'}$. The subgraph induced by the node set $A \subseteq V$ is $G[A] = (A, E')$, where $E' = \{e \in E : e \subseteq A\}$. Similarly, the subgraph induced by an edge set $F \subseteq E$ is $G[F] = (V', F)$, where $V' = \{u \in e : e \in F\}$. A graph G is *d-degenerate* if every subgraph $G' \subseteq G$ contains a node with degree at most d. Every graph is trivially $(n-1)$-degenerate. We note that degeneracy is within a constant factor of an another widely-used graph parameter *arboricity*, that is, any graph with arboricity a has degeneracy $a \leq d \leq 2a - 1$; see e.g. Barenboim and Elkin [3].

CONGEST and broadcast CONGEST. The CONGEST model is a variant of classical LOCAL model of distributed computation with additional constraints on communication bandwidth [27]. The distributed system is represented as a network $G = (V, E)$, where each node $v \in V$ executes the same algorithm in synchronous rounds, and the nodes collaborate to solve a graph problem with input G. Each round, all nodes

1. perform an unlimited amount of local computation,
2. send a possibly different $O(\log n)$-bit message to each of their neighbours, and
3. receive the messages sent to them.

The time measure is the number of synchronous rounds required. Each node is assumed to have a unique identifiers from the set $\{0, \ldots, \text{poly}(n)\}$.

The broadcast CONGEST is a weaker version of CONGEST, with the additional constraint that all nodes have to send the same message to each of their neighbours.

4 Finding paths, cycles and trees

Representative families. The general cycle and path detection algorithms are based on *representative families* [24, 14]. For a basic intuition, consider a setting where we have a collection of objects (in this case, sets) of size p, and we want to extend them by adding at most q more elements. Representative families allow us to 'compress' our collection of objects so that if a member of the original collection could be extended by specific q elements, then the compressed collection also contains a member that can be extended by the same elements. Indeed, while we only use the set family version of this theory, it can be extended to *matroids* [18].

▶ **Definition 1.** Let $\mathcal{A} \subseteq 2^{[n]}$. We say that $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ is *q-representative* for $\mathcal{A}$ if for each $B \subseteq [n]$ with $|B| \leq q$, there is a set $A \in \mathcal{A}$ with $A \cap B = \emptyset$ if and only if there is $\widehat{A} \in \widehat{\mathcal{A}}$ with $\widehat{A} \cap B = \emptyset$.

▶ **Theorem 2** ([14]). *Let $\mathcal{A} \subseteq 2^{[n]}$ consist of sets of size at most p. Then there is a q-representative $\widehat{\mathcal{A}} \subseteq \mathcal{A}$ with $|\widehat{\mathcal{A}}| \leq \binom{p+q}{p}$.*

For our purposes it is sufficient to know that small representative families exist; however, slightly worse bounds with corresponding efficient algorithms are known [23, 18, 14].

Finally, we will use the following simple fact about representative families:

▶ **Lemma 3.** *Let $\mathcal{C} \subseteq \mathcal{B} \subseteq \mathcal{A} \subseteq 2^{[n]}$. 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 $\mathcal{A}$.*

Together Theorem 2 and Lemma 3 imply that any inclusion-minimal q-representative family for a family of sets of size at most p has size at most $\binom{p+q}{p}$.

Finding paths. We start by giving a simple path detection algorithm for the CONGEST model, using a well-known technique from fixed-parameter algorithmics [24]. Intuitively, the idea is to start with the trivial path detection algorithm that iteratively propagates full information about paths of length at most $\ell = 1, 2, \ldots, k$ in k phases. This will be quite slow if there are lots of such paths; however, at each step, we construct a representative family for the current set of paths to ensure that we only forward the essential ones.

Let k be fixed, and define for $v \in V$ and $\ell \in \{1, 2, \ldots, k\}$ the set family

$$\mathcal{P}_{v,\ell} = \left\{U \subseteq V : \text{there is an } \ell\text{-path with node set } U \text{ that ends at } v\right\}.$$

By definition, we have that

$$\mathcal{P}_{v,\ell} = \bigcup_{u \in N(v)} \left\{\{v\} \cup U : U \in \mathcal{P}_{u,\ell-1} \text{ and } v \notin U\right\}.$$

The basic idea of the algorithm is that instead of explicitly constructing the families $\mathcal{P}_{v,1}, \mathcal{P}_{v,2} \ldots, \mathcal{P}_{v,k}$, we iteratively construct families $\widehat{\mathcal{P}}_{v,1}, \widehat{\mathcal{P}}_{v,2} \ldots, \widehat{\mathcal{P}}_{v,k}$, where $\widehat{\mathcal{P}}_{v,\ell}$ is a minimal $(k-\ell)$-representative family for $\mathcal{P}_{v,\ell}$.

Specifically, the algorithm proceeds as follows:

1. For $\ell = 1$, we have that $\mathcal{P}_{v,1} = \{\{u, v\} : u \in N(v)\}$. Each node v can obtain full information about $N(v)$ in single round, and then compute $\widehat{\mathcal{P}}_{v,1}$ locally.
2. For $\ell \geq 2$, after the families $\widehat{\mathcal{P}}_{v,\ell-1}$ have been constructed, each node $v \in V$ broadcasts the family $\widehat{\mathcal{P}}_{v,\ell-1}$ to all of its neighbours. Since $\widehat{\mathcal{P}}_{v,\ell-1}$ is a minimal $(k-\ell+1)$-representative family for $\mathcal{P}_{v,\ell-1}$, and $\mathcal{P}_{v,\ell-1}$ consists of sets of size ℓ, we have that $|\widehat{\mathcal{P}}_{v,\ell-1}| \leq \binom{k+1}{\ell-1} \leq 2^{k+1}$. In particular, $\widehat{\mathcal{P}}_{v,\ell-1}$ can be encoded using $O\left(k\binom{k+1}{\ell-1} \log n\right)$ bits, and broadcasting it to all neighbours can be done in $O\left(k\binom{k+1}{\ell-1}\right)$ rounds.
3. Each node $v \in V$, after having received $\widehat{\mathcal{P}}_{u,\ell-1}$ for each $u \in N(v)$, locally constructs the set

$$\mathcal{P}'_{v,\ell} = \bigcup_{u \in N(v)} \left\{\{v\} \cup U : U \in \widehat{\mathcal{P}}_{u,\ell-1} \text{ and } v \notin U\right\}.$$

   We observe that $\mathcal{P}'_{v,\ell}$ is $(k-\ell)$-representative for $\mathcal{P}_{v,\ell}$. If $W \subseteq V$ is a set of size $k-\ell$ that does not intersect some $U \in \mathcal{P}_{v,\ell}$, then there is some $u \in N(v)$ and $U' \in \mathcal{P}_{u,\ell-1}$ such that $W \cup \{v\}$ does not intersect U'. Since $\widehat{\mathcal{P}}_{u,\ell-1}$ is $(k-\ell+1)$-representative for $\mathcal{P}_{u,\ell-1}$, there is $R \in \widehat{\mathcal{P}}_{u,\ell-1}$ such that R does not intersect $W \cup \{v\}$, and thus $R \cup \{v\} \in \mathcal{P}'_{v,\ell}$ does not intersect W. Thus, computing a minimal $(k-\ell)$-representative family $\widehat{\mathcal{P}}_{v,\ell}$ for $\mathcal{P}'_{v,\ell}$ gives a $(k-\ell)$-representative family for $\mathcal{P}_{v,\ell}$ by Lemma 3.

Clearly, there is a k-path terminating at v if and only if $\widehat{\mathcal{P}}_{v,k}$ is non-empty. There are k phases corresponding to $\ell = 1, 2, \ldots, k$ in the algorithm, and each of these phases runs in

$O\big(k\binom{k+1}{\ell-1}\big)$ rounds, where the hidden constant does not depend on ℓ. Since it holds that is

$$\sum_{\ell=1}^{k} k\binom{k+1}{\ell-1} = k\sum_{\ell=1}^{k}\binom{k+1}{\ell-1} \leq k2^k\,,$$

the total running time is $O(k2^k)$.

Finally, let us observe that it is easy to modify this algorithm to *find* a k-path, in the sense that each node $v \in V$ that is an endpoint of a k-path has full knowledge of a single k-path: we annotate each set $U \in \widehat{\mathcal{P}}_{v,\ell}$ with a 'witness' ℓ-path with node set U terminating at v. This can be done without increasing the asymptotic communication cost.

▶ **Theorem 4.** *Finding k-paths can be done in $O(k2^k)$ rounds in the broadcast* CONGEST *model.*

Finding cycles. We first note that it is easy to adapt the k-path algorithm to detect k-cycles that contain a fixed node $w \in V$. We modify the definition of $\mathcal{P}_{v,\ell}$ to require that the paths have w as a starting point; otherwise the algorithm proceeds as before. If any neighbour of w detects a $(k-1)$-path starting from w, then there is a k-cycle containing w; likewise, if there is a k-cycle containing w, then some neighbour of w will detect a $(k-1)$-path starting from w. Running this cycle detection algorithm for all choices of the starting node in parallel gives allows us to detect k-cycles in time $O(k2^k n)$:

▶ **Theorem 5.** *In the broadcast* CONGEST *model,*
1. *finding k-cycles containing a fixed node $w \in V$ can be done in $O(k2^k)$ rounds, and*
2. *finding k-cycles can be done in $O(k2^k n)$ rounds.*

Finding trees. We now show how to extend the path detection algorithm to any target graph H that is acyclic. Let H be a tree on k nodes. Fix an arbitrary node of H as a root, and identify the nodes of H with $1, 2, \ldots, k$ so that if i is a descendant of j then $j > i$. In particular, node k is the root. Denote by $H[i]$ the subtree of H rooted at node i, and denote by s_i the number of nodes in $H[i]$.

For $v \in V$ and $i \in \{1, 2, \ldots, k\}$, we define the set family

$$\mathcal{T}_{v,i} = \big\{U \subseteq V\colon \text{there is a copy of } H[i] \text{ with node set } U \text{ and root } v\big\}\,.$$

Again, if we know $\mathcal{T}_{u,j}$ for all $u \in N(v)$ and and $j < i$, we can compute $\mathcal{T}_{v,i}$. Let $c_1, c_2, \ldots, c_\ell$ be the children of i in H. Then $\mathcal{T}_{v,i}$ contains exactly the sets of form

$$\{v\} \cup U_1 \cup U_2 \cup \cdots \cup U_\ell\,,$$

where $U_x \in \mathcal{T}_{u,c_x}$ for some $u \in N(v)$ and $U_x \cap U_y = \emptyset$ for all $x \neq y$.

Similarly to the path algorithm, for $i = 1, 2, \ldots, k$, we iteratively construct $(k - s_i)$-representative family $\widehat{\mathcal{T}}_{v,i}$ for $\mathcal{T}_{v,i}$:
1. If i is a leaf in H, then $\widehat{\mathcal{T}}_{v,i} = \mathcal{T}_{v,i} = \{\{v\}\}$.
2. If i is an internal node in H with children $c_1, c_2, \ldots, c_\ell$, then we construct $\mathcal{T}'_{v,i}$ by taking all sets of form

   $$\{v\} \cup U_1 \cup U_2 \cup \cdots \cup U_\ell\,,$$

   where $U_x \in \widehat{\mathcal{T}}_{u,c_x}$ for $u \in N(v)$ and $U_x \cap U_y = \emptyset$ for all $x \neq y$, and compute a minimal $(k - s_i)$-representative family $\widehat{\mathcal{T}}_{v,i}$ for $\mathcal{T}'_{v,i}$.

We observe that after step i, each node v knows if there is a copy of $H[i]$ that is rooted at v, which implies the correctness of the algorithm. Time complexity follows by the same arguments as in the path detection algorithm. To summarise:

▶ **Theorem 6.** *For any tree H on k nodes, H-subgraph detection can be done in $O(k2^k)$ rounds in the broadcast* CONGEST *model.*

Finding pseudotrees. Recall that a pseudotree is a connected graph that has at most one cycle, that is, a graph that consist of a tree and at most one additional edge. Let H be a pseudotree on k vertices, and let us assume that it has a cycle. Let $e = \{u_1, u_2\}$ be an arbitrary edge on the cycle.

We slightly modify our tree detection algorithm to find a copy of H as follows. Consider the tree H' obtained from H by deleting the edge e. We identify the nodes of H' with $1, 2, \ldots, k$ as before, using u_2 as the root node k; let us assume that u_1 is identified with j.

The basic idea is to modify the tree detection algorithm to keep track which nodes of the input graph can play the role of j in a copy of H'. Specifically, when constructing the sets $\widehat{\mathcal{T}}_{v,i}$, we keep track of which node plays the role of j in the copies of $H'[i]$ we have found so far. This requires the following changes to the tree detection algorithm:

- For i that is not on the unique path from j to k, no changes are required.
- For the step corresponding to j, each node $v \in V$ performs the step as before, but marks itself as the node playing the role of j when broadcasting the constructed representative family $\widehat{\mathcal{T}}_{v,j}$.
- For i that is on the path from j to k, each node $v \in V$ keeps track of $\widehat{\mathcal{T}}_{v,i}$ separately for each possible choice of j it sees.

After the algorithm is finished, all nodes check if they see a copy of H' where the node j is one of their neighbours, which happens if and only if a copy of H exists in the graph. At each step, each node has to broadcast at most n copies of the representative family, giving us the following:

▶ **Theorem 7.** *For any pseudotree H on k nodes, H-subgraph detection can be done in $O(k2^k n)$ rounds in the broadcast* CONGEST *model.*

5 Enumerating cliques and short cycles in degenerate graphs

Acyclic orientations with bounded outdegree. Our algorithms for enumerating cliques and cycles in sparse graphs exploit the fact that degenerate graphs have acyclic orientations with bounded outdegree. Once we have such an orientation, the nodes can co-ordinate how to distribute the information about the edges present in the input graph, which avoids having too much congestion over a single communication link.

For any $\alpha \in \mathbb{N}$, we say that σ is an α-bounded orientation [10] of $G = (V, E)$ if

1. every node $v \in V$ has $\mathrm{outdeg}_\sigma(v) \leq \alpha$, and
2. the orientation σ is acyclic.

For brevity, we call such orientations simply α-orientations. The class of degenerate graphs can be characterised by the existence of such orientations.

▶ **Lemma 8.** *A graph G is d-degenerate if and only if there exists a d-orientation of G.*

Barenboim and Elkin [2] gave efficient distributed algorithms for computing such orientations under the name Nash–Williams forests-decompositions [25]. For the sake of completeness, we formulate more or less the same algorithm here. Barenboim and Elkin [2]

also show how to compute such orientations even without knowing either the degeneracy, or alternatively, the number of nodes in the network in $O(\log n)$ rounds using messages of size $O(\log n)$.

▶ **Lemma 9.** *If G is d-degenerate, then it has at most nd edges.*

Proof. Consider the d-orientation given by Lemma 8. Counting the outgoing edges yields

$$|E| = \sum_{v \in V} \text{outdeg}(v) \leq nd.$$ ◀

Let G be a d-degenerate graph and $C > 2$ be a constant. For all $i \geq 0$, define iteratively the node sets $V_0, V_1, V_2, \ldots$ as follows:

$$V_0 = V\,,$$
$$V_{i+1} = \{v \in V_i : \deg(v) > Cd\}\,.$$

We note that each node can compute $i_v = \max\{i \in \mathbb{N} \colon v \in V_i\}$ in i_v rounds.

▶ **Lemma 10.** *We have that $|V_{i+1}| \leq 2/C \cdot |V_i|$.*

Proof. Suppose the opposite holds, that is, there are $h > 2/C \cdot |V_i|$ nodes $v \in V_i$ with $\deg(v) > Cd$. As $G[V_i] \subseteq G$ is d-degenerate, counting the number of edges incident to these h nodes yields that there are at least

$$\frac{Cdh}{2} > \frac{2 \cdot |V_i| Cd}{2C} = |V_i| \cdot d$$

edges, which contradicts Lemma 9. ◀

By the above lemma, we get that every iteration we lose a constant fraction $1/c$ of nodes, where $c = C/2$. Thus, $|V_i| \leq n/c^i$ and after $r = \log n / \log c + 1 = \log_c n$ iterations we have

$$|V_r| = \frac{n}{c^{\log_c n + 1}} \leq 1.$$

We can now compute an acyclic orientation of G using the above scheme. Each node $v \in V_i \setminus V_{i+1}$ removed in iteration i orients any edge $\{v, u\}$ with $u \in V_{i+1}$ towards u. Clearly there will be at most Cd edges pointing away from v.

▶ **Lemma 11** ([2])**.** *Let G be a d-degenerate graph. For any constant $\varepsilon > 0$, we can compute a $(2d + \varepsilon)$-orientation of G in $O(\log n)$ rounds in the broadcast* CONGEST *model.*

Enumerating cliques. We start with a k-clique enumeration algorithm that works in d-degenerate graphs for any $k \leq d$. Note that the scenario $k > d$ is fatuous, as a d-degenerate graph cannot contain a clique with more than d nodes. Let $G = (V, E)$ be the input graph. The algorithm is as follows:

1. Compute the α-orientation of G for $\alpha \in \Theta(d)$.
2. Each $v \in V$ broadcasts the endpoints $N_{\text{out}}(v)$ of outgoing edges to all its neighbours.
3. Each $v \in V$ locally constructs the induced subgraph $G[F]$, where

$$F = \{\{u, w\} : u \in N^+(v), w \in N_{\text{out}}(u)\}.$$

4. Each $v \in V$ outputs all k-cliques in $G[F]$.

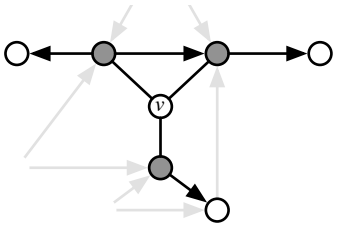


Figure 1 Example of the information gathered by the clique detection algorithm. Once an α-orientation of the graph has been computed, the algorithm ensures that node v obtains all outgoing edges of its neighbours (black edges). Neighbours of v are are dark gray and the edges not observed by v are gray. Here, node v detects the triangle its part of.

Figure 1 illustrates the information gathered by the above algorithm.

In order to analyse the algorithm, first observe that the algorithm clearly only outputs k-cliques that are present in G. We now argue that if G contains a k-clique, then some node will list it in the output. Let $K \subseteq V$ be a k-clique in G. Note that $G[K]$ must contain a sink node $v \in K$ with respect to the α-orientation σ: if such a sink node does not exist, then every node in $G[K]$ would have positive outdegree implying that σ is not acyclic, which is absurd. Hence, the sink node $v \in K$ will receive the set $N_{\text{out}}(u)$ from every $u \in K$. In particular, we have

$$K \subseteq \bigcup_{u \in N(v)} N_{\text{out}}(u).$$

Therefore, the sink node v detects the clique K in the subgraph H it constructs in the third step, and thus, will list it in its output.

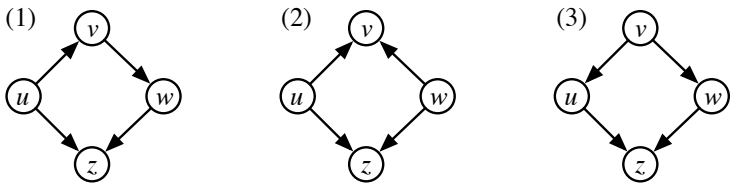
It remains to analyse the time complexity of the algorithm. By Lemma 11, the first step takes $O(\log n)$ rounds. The second step takes $O(\alpha)$ rounds, as broadcasting a single identifier requires $O(\log n)$ bits to be transmitted along each edge. Since there are at most α such identifiers to be broadcast (as there are at most α outgoing edges) and $\alpha \in O(d)$, we get that this step takes $O(d)$ time. The remaining steps require no communication, and hence, all nodes declare their output within $O(\log n + d)$ rounds.

▶ **Theorem 12.** *Let $k \leq d$. In any d-degenerate graph G, enumerating k-cliques takes $O(d + \log n)$ rounds in the broadcast* CONGEST *model.*

As triangles are 3-cliques, we immediately get the following corollary.

▶ **Corollary 13.** *In any d-degenerate graph G, enumerating triangles takes $O(d + \log n)$ rounds in the broadcast* CONGEST *model.*

Enumerating 4-cycles. In the case of 4-cycles, it turns out that we can use the same algorithm as for enumerating cliques, except we check for the existence of a 4-cycle in the locally constructed subgraph $G[F]$ instead of a k-clique. Let $C = \{u, v, w, z\}$ be a 4-cycle in G. As before, we first obtain α-orientation σ of the d-degenerate input graph G using Lemma 11. Since the orientation is acyclic, the cycle graph $G[C]$ must have at least one source and a sink in the orientation. In particular, there are three possible cases (up to isomorphisms):



To show that some node detects the cycle C, we consider each of these three cases. In each case, observe that u sends v the set $N_{\text{out}}(u)$ and w sends the set $N_{\text{out}}(w)$ to v. Thus, node v learns about the edges $\{u, z\}$ and $\{w, z\}$ as $z \in N_{\text{out}}(u) \cap N_{\text{out}}(w)$. Since v trivially knows about the edges $\{u, v\}$ and $\{v, w\}$, we get that in each case node v learns about all edges in C and lists C in its output. Note that if we want that each 4-cycle is listed by exactly one node, the nodes can detect if multiple nodes would be listing the same cycle C and output C only if they are the node with smallest identifier having knowledge about it.

As the algorithm communicates the same information as the clique detection algorithm given before, we get that the time complexity of detecting 4-cycles is the same.

▶ **Theorem 14.** *In any d-degenerate graph G, enumerating 4-cycles takes $O(d+\log n)$ rounds in the broadcast* CONGEST *model.*

Enumerating 5-cycles. In order to enumerate 5-cycles, we use the same underlying idea, but now nodes also communicate directed outgoing paths of length 2 instead of just single outgoing edges. With this modification in mind, the algorithm is as follows:

1. Compute the α-orientation of G for $\alpha \in \Theta(d)$.
2. Each $v \in G$ broadcasts its endpoints $N_{\text{out}}(v)$ of outgoing edges to all its neighbours.
3. Each $v \in G$ broadcasts the set of outgoing length-2 paths

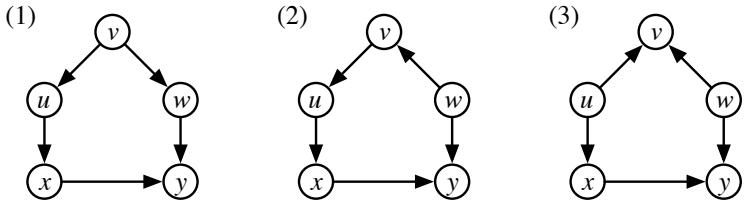
$$L(v) = \{\{u, w\} : u \in N_{\text{out}}(v), w \in N_{\text{out}}(u)\}$$

   to all its neighbours.
4. Each $v \in G$ locally constructs the subgraph $G[F]$, where

$$F = \big\{\{u, w\} : u \in N^+(v), w \in N_{\text{out}}(u)\big\} \cup \bigcup_{u \in N_{\text{out}}(v)} L(u).$$

5. Each $v \in V$ outputs all 5-cycles in $G[F]$.

Let $C = \{u, v, w, x, y\}$ be a 5-cycle in G. To see that the algorithm outputs the cycle C, it suffices to show that some node $v \in C$ learns about all the edges in C. As the α-orientation is acyclic, the cycle C can be oriented in the following three ways (up to isomorphisms):



To see that this is the case, one can readily observe that the length of the longest directed path in an acyclically oriented 5-cycle has to be either 2, 3 or 4, and the length of this path determines the orientation of all other edges.

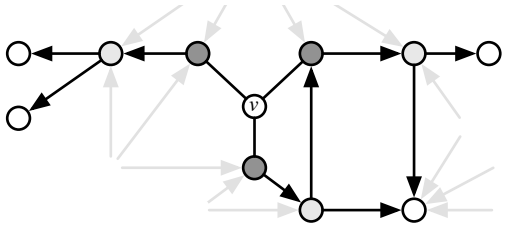


Figure 2 Example of information the nodes gather when executing the 5-cycle detection algorithm. Once an α-orientation of the graph has been computed, the algorithm ensures that node v obtains all outgoing length-1 and length-2 paths (black directed edges) of its neighbours. Neighbours of v are are dark gray, nodes at distance two are light gray and nodes at distance three are white.

We argue that node v detects the cycle C in all cases. Observe that node x broadcasts $N_{\text{out}}(x)$ to u in the second step and node u broadcasts $N_{\text{out}}(u)$ and $L(u)$ to v in the second and third steps, respectively. Since $y \in N_{\text{out}}(x)$, we have $\{x, y\} \in L(u)$ and $x \in N_{\text{out}}(u)$. Therefore, v learns about the path (u, x, y). Since w broadcasts $N_{\text{out}}(w)$ to v, node v also learns about the edge $\{w, y\}$. Since node v can trivially detect the edges $\{u, v\}$ and $\{v, w\}$, it follows that node v detects the cycle C in the final step.

The time complexity of the first step is again $O(\log n)$. The second step consists of broadcasting the set of up to α identifiers, which takes $O(\alpha)$ rounds as before. In the third step, each node $v \in G$ broadcasts $L(v)$ which may contain up to $O(\alpha^2)$ edges. Thus, the third step takes $O(\alpha^2)$ rounds. No communication occurs in the final steps of the algorithm, which yields that the total time complexity of the algorithm is $O(\log n + \alpha^2)$. As we can choose $\alpha \in \Theta(d)$, we obtain the following result.

▶ **Theorem 15.** *In any d-degenerate graph G, enumerating 5-cycles takes $O(d^2 + \log n)$ rounds in the broadcast* CONGEST *model.*

6 Lower bounds

Lower bounds for long cycles. We first prove our lower bounds for the detection of long cycles:

▶ **Theorem 16.** *In the* CONGEST *model,*

1. *finding k-cycles for even k requires $\Omega(n^{1/2}/\log n)$ rounds, and*
2. *there is no algorithm for finding k-cycles for $k \geq 6$ that runs in $O(f(d)n^{\delta})$ rounds on d-degenerate graphs for any function f and $\delta < 1/2$.*

Proof. We use a slight modification of the cycle detection lower bound construction of Drucker et al. [13]; that is, we prove the lower bound by a reduction to known set disjointness lower bounds in two-player communication complexity.

Specifically, given sets A and B over M-element universe, we construct a n-node graph $G_{A,B} = (V_A \cup V_B, E)$ such that (1) $G[V_A]$ depends only on A, (2) $G[V_B]$ depends only on B, (3) there are at most C edges crossing the cut (V_A, V_B), and (4) $G_{A,B}$ contains a cycle of length k if and only if A and B are non-disjoint. This allows us to use any k-cycle detection to solve 2-party set disjointness; Alice simulates all nodes in V_A, Bob simulates nodes in V_B, and messages over edges crossing the cut (V_A, V_B) are sent between players. By known lower bounds, $\Omega(M)$ bits have to be sent over the cut, giving lower bound of $\Omega(M/C \log n)$ rounds in the CONGEST model.

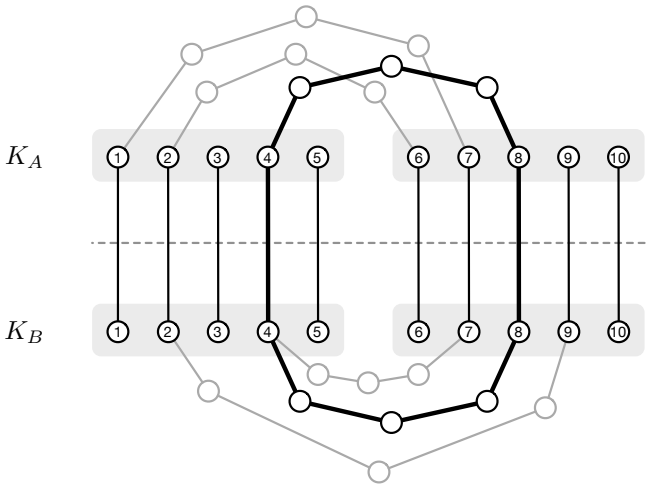


Figure 3 Example of the lower bound construction for $k = 8$, $N = 5$ and $M = 25$. A 8-cycle is highlighted in black; note that a 8-cycle can occur only when the corresponding paths are present both in K_A and K_B.

We show that for any $k \geq 6$ and all $N \in \mathbb{N}$, there is a construction satisfying the above with parameters $M \in \Theta(N^2)$, $n \in \Theta(N^2)$ and $C = 2N$, with the additional property that any graph obtained from this construction is 2-degenerate. This implies both of the claims.

Let $k \geq 6$ and $N \in \mathbb{N}$ be fixed, and let $A, B \subseteq 2^{[N^2]}$ be a set disjointness instance. Let $\ell_1 = \lfloor k/2 \rfloor$ and $\ell_2 = \lceil k/2 \rceil$. We take two disjoint copies K_A and K_B of a complete bipartite graph $K_{N,N}$ on $2N$ nodes; assume that the nodes of the both copies are labelled in a consistent manner with integers $1, 2, \ldots, 2N$, and that the edges are likewise consistently labelled with the elements of $2^{[N^2]}$. We now finish the construction as follows:

1. We connect nodes labelled with the same integer in K_A and K_B with an edge.
2. We remove each edge in K_A if the corresponding element in $2^{[N^2]}$ is not in A, and we remove each edge in K_B if the corresponding element in $2^{[N^2]}$ is not in B.
3. We replace all remaining edges in K_A with $(\ell_1 - 1)$-paths, and all remaining edges in K_B with $(\ell_2 - 1)$-paths.

It is now easy to verify that there are no k-cycles remaining inside either K_A or K_B, and there is a k-cycle if and only if $A \cap B \neq \emptyset$. The final graph has at most $(k-4)N^2 + 4N$ nodes and at most $(k-2)N^2 + 2N$ edges; the cut between K_A and K_B has size $2N$. Finally, it is easy to see that the graph has degeneracy 2. For each path added in step (3), we pick an internal node and orient all edges away from it, and orient the cut edges between K_A and K_B arbitrarily; the resulting orientation has maximum outdegree two. ◀

Lower bounds for short cycles. We observe that the lower bounds given by Drucker et al. [13] imply that finding 4-cycles and 5-cycles in d-degenerate graphs requires $\Omega(d/\log n)$ rounds: a careful examination shows that their lower bound graph for 4-cycles has degeneracy $\Theta(n^{1/2})$ and the lower bound graph for 5-cycles has degeneracy $\Theta(n)$. Thus, we have the following:

▶ **Theorem 17.** *Finding 4-cycles and 5-cycles in d-degenerate graphs requires $\Omega(d/\log n)$ rounds in the* CONGEST *model.*

7 Extensions to the supported CONGEST

The supported CONGEST model. Finally, we discuss how our results can be extended to the supported CONGEST proposed by Schmid and Suomela [29] for studying distributed algorithms in the context of *software defined networking* (SDN).

In the supported CONGEST model, the underlying physical network topology is represented by a *support graph* $G = (V, E)$ and the actual *input graph* H for the distributed algorithm is an arbitrary subgraph of the support. The input graph $H = (U, F)$ inherits the identifiers of the support graph and can be seen as the current logical state of the network. Each node v in the network G is aware of the full topological information about G, but node v has only local information about the current logical state of the network, that is, the actual input graph H. That is, initially node v will know only which of the edges incident to it in G are also present in H. Note that supported CONGEST can be seen as a relaxation of the congested clique model: in the case of the congested clique model, the support graph is a clique.

While many symmetry breaking problems are trivial in the supported CONGEST model, the model remains interesting from the perspective of subgraph detection. Even though all nodes will a priori know that any edge $\{u, v\}$ not present in G will also not be in H, only nodes u and v initially know whether $\{u, v\} \in F$ as well. Thus, the difficulty now becomes verifying which of the possible subgraphs in G remain in H.

Subgraph enumeration in sparse support graphs. We can modify the algorithms given in Section 5 to run faster in the supported CONGEST model, in the case where the support graph has bounded degeneracy. First, all nodes can locally determine the degeneracy of the support graph G and compute (the same) d-orientation σ of G without any communication. The orientation σ restricted to the input graph H will also be an d-orientation of the input H. This saves the additive $O(\log n)$ term in the running time. Moreover, when the nodes communicate their outgoing edges, as every node $v \in V$ knows which outgoing edges of σ *can be* present incident to $u \in V$, it suffices that each node u broadcasts d-length bit string encoding which edges of G are present in the input H. With these modifications in mind, we get the following result.

▶ **Theorem 18.** *In any d-degenerate support graph G in the supported* CONGEST *model,*

1. *enumerating k-cliques can be done in $O(d/\log n)$ rounds for any k,*
2. *enumerating 4-cycles can be done in $O(d/\log n)$ rounds, and*
3. *enumerating 5-cycles can be done in $O(d^2/\log n)$ rounds.*

In particular, in the case that the support graph has degeneracy $O(\log n)$, we can enumerate cliques and 4-cycles in constant number of rounds in the supported CONGEST model.

Cycle detection lower bounds. Finally, we note that all lower bounds from the present work as well as those by Drucker et al. [13] hold in the supported CONGEST. All these lower bounds are obtained by starting from a fixed base graph G, and removing certain edges to obtain a graph G' that encodes a set disjointness instance. Taking the base graph G as the support graph and G' as the input graph yields identical lower bounds for the supported CONGEST.

8 Conclusions

Subgraph detection. We note that there still remain major open questions regarding subgraph detection in the CONGEST model:

- What is the complexity of general subgraph detection? In particular, can the subgraph detection problem solved in linear number of rounds for any constant-size target graph H, or are the target graphs that require a superlinear number of rounds? Note that k-cliques, which appear to be the most difficult case for centralised subgraph detection, can be trivially detected in $O(n)$ rounds for any k.
- What is the precise complexity of detecting even-length cycles? Is there an algorithm matching the $\tilde{\Omega}(n^{1/2})$ lower bound we give, or can this lower bound be improved?

Fixed-parameter techniques. We remark that the many algorithmic techniques from fixed-parameter cycle and path detection algorithms seem to translate very naturally to the distributed limited bandwidth setting, as they effectively either (a) partition the task at hand into multiple independent instances of an easier task (e.g. colour-coding), or (b) compress the intermediate results of the computation (e.g. representative families). As noted before, the seminal colour-coding technique of Alon et al. [1] has been used by various authors for distributed algorithms [9, 16, 17]; indeed, the results in the present work can also be derived via colour-coding, albeit with slightly worse dependence on k, by an easy modification of the congested clique cycle detection algorithm of Censor-Hillel et al. [9]. We also expect that the $O(k2^k)$ round running time for path detection could be improved with randomisation, for example by using algebraic *sieving* techniques [6].

More generally, such fixed-parameter techniques are applicable in the centralised setting beyond subgraph detection, and we expect this to be the case also in the distributed setting. For example, it seems likely that constant-round distributed algorithms for the *graph motif* problem can be obtained either via colour-coding or algebraic sieving [6, 5]. Indeed, this general line of research may even have practical relevance, as evidenced by the efficient parallel implementation of fixed-parameter graph motif algorithms by Björklund et al. [7].

Acknowledgements. We thank Juho Hirvonen, Dennis Olivetti, Rotem Oshman, Chris Purcell, Stefan Schmid and Jukka Suomela for valuable comments and discussions.

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

2 Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. *Distributed Computing*, 22(5-6):363–379, 2010. doi:10.1007/s00446-009-0088-2.

3 Leonid Barenboim and Michael Elkin. *Distributed Graph Coloring: Fundamentals and Recent Developments*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2013. doi:10.2200/S00520ED1V01Y201307DCT011.

4 Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The locality of distributed symmetry breaking. *J. ACM*, 63(3):20:1–20:45, 2016. doi:10.1145/2903137.

5 Nadja Betzler, Michael R. Fellows, Christian Komusiewicz, and Rolf Niedermeier. Parameterized algorithms and hardness results for some graph motif problems. In *Proc. 19th Annual Symposium on Combinatorial Pattern Matching (CPM 2008)*, pages 31–43. Springer, 2008.

6 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Narrow sieves for parameterized paths and packings. *J. Comput. Syst. Sci.*, 87:119–139, 2017. doi:10.1016/j.jcss.2017.03.003.

7 Andreas Björklund, Petteri Kaski, Lukasz Kowalik, and Juho Lauri. Engineering motif search for large graphs. In Ulrik Brandes and David Eppstein, editors, *Proceedings of the Seventeenth Workshop on Algorithm Engineering and Experiments, ALENEX 2015, San Diego, CA, USA, January 5, 2015*, pages 104–118. SIAM, 2015. doi:10.1137/1.9781611973754.10.

8 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, volume 9888 of *Lecture Notes in Computer Science*, pages 43–56. Springer, 2016. doi:10.1007/978-3-662-53426-7_4.

9 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Proc. ACM Symposium on Principles of Distributed Computing (PODC 2015)*, pages 143–152, 2015.

10 Marek Chrobak and David Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theor. Comput. Sci.*, 86(2):243–266, 1991. doi:10.1016/0304-3975(91)90020-3.

11 Danny Dolev, Christoph Lenzen, and Shir Peled. "tri, tri again": Finding triangles and small subgraphs in a distributed setting - (extended abstract). In Marcos K. Aguilera, editor, *Distributed Computing - 26th International Symposium, DISC 2012, Salvador, Brazil, October 16-18, 2012. Proceedings*, volume 7611 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2012. doi:10.1007/978-3-642-33651-5_14.

12 Rodney G. Downey and Michael R. Fellows. Parameterized computational feasibility. In *Feasible Mathematics II*, pages 219–244, 1994. doi:10.1007/978-1-4612-2566-9_7.

13 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 367–376. ACM, 2014. doi:10.1145/2611462.2611493.

14 P. Erdős, A. Hajnal, and J. W. Moon. A problem in graph theory. *The American Mathematical Monthly*, 71(10):1107–1110, 1964. doi:10.2307/2311408.

15 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Dennis Olivetti Pedro Montealegre, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three notes on distributed property testing. In *Proc. 31st International Symposium on Distributed Computing (DISC 2017)*, 2017.

16 Guy Even, Reut Levi, and Moti Medina. Faster and simpler distributed algorithms for testing and correcting graph properties in the CONGEST-model, 2017. arXiv:1705.04898 [cs.DC].

17 Orr Fischer, Tzlil Gonen, and Rotem Oshman. Distributed property testing for subgraph-freeness revisited, 2017. arXiv:1705.04033 [cs.DS].

18 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 142–151, 2014.

19 Pierre Fraigniaud, Pedro Montealegre, Dennis Olivetti, Ivan Rapaport, and Ioan Todinca. Distributed subgraph detection, 2017. arXiv:1706.03996 [cs.DC].

20 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. Distributed testing of excluded subgraphs. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceed-*

ings, volume 9888 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2016. doi:10.1007/978-3-662-53426-7_25.

21 Taisuke Izumi and François Le Gall. Triangle finding and listing in CONGEST networks. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 381–389. ACM, 2017. doi:10.1145/3087801.3087811.

22 Janne H. Korhonen and Joel Rybicki. Deterministic subgraph detection in broadcast CONGEST, 2017. arXiv:1705.10195 [cs.DC].

23 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009.

24 Burkhard Monien. How to find long paths efficiently. *North-Holland Mathematics Studies*, 109:239–254, 1985.

25 Crispin Nash-Williams. Decomposition of finite graphs into forests. *Journal of the London Mathematical Society*, s1-39(1):12–12, 1964. doi:10.1112/jlms/s1-39.1.12.

26 Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Tight bounds for distributed graph computations, 2016. arXiv:1602.08481 [cs.DC].

27 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2000.

28 Oleg Pikhurko. A note on the Turán function of even cycles. *Proceedings of the Americal Mathematical Society*, 140:3687–3692, 2012. doi:10.1090/S0002-9939-2012-11274-2.

29 Stefan Schmid and Jukka Suomela. Exploiting locality in distributed SDN control. In Nate Foster and Rob Sherwood, editors, *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, The Chinese University of Hong Kong, Hong Kong, China, Friday, August 16, 2013*, pages 121–126. ACM, 2013. doi:10.1145/2491185.2491198.