# The Complexity Landscape of Dynamic Distributed Subgraph Finding

**Yi-Jun Chang**
National University of Singapore, Singapore

**Lyuting Chen**
National University of Singapore, Singapore

**Yanyu Chen**
National University of Singapore, Singapore

**Gopinath Mishra**
National University of Singapore, Singapore

**Mingyang Yang**
National University of Singapore, Singapore

## Abstract

Bonne and Censor-Hillel (ICALP 2019) initiated the study of distributed subgraph finding in dynamic networks of limited bandwidth. For the case where the target subgraph is a clique, they determined the tight bandwidth complexity bounds in nearly all settings. However, several open questions remain, and very little is known about finding subgraphs beyond cliques. In this work, we consider these questions and explore subgraphs beyond cliques in the *deterministic* setting.

For finding cliques, we establish an $\Omega(\log\log n)$ bandwidth lower bound for one-round membership-detection under edge insertions only and an $\Omega(\log\log\log n)$ bandwidth lower bound for one-round detection under both edge insertions and node insertions. Moreover, we demonstrate new algorithms to show that our lower bounds are *tight* in bounded-degree networks when the target subgraph is a triangle. Prior to our work, no lower bounds were known for these problems.

For finding subgraphs beyond cliques, we present a *complete characterization* of the bandwidth complexity of the membership-listing problem for every target subgraph, every number of rounds, and every type of topological change: node insertions, node deletions, edge insertions, and edge deletions. We also show partial characterizations for one-round membership-detection and listing.

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** Distributed algorithms, dynamic algorithms, subgraph finding

**Digital Object Identifier** 10.4230/LIPIcs.DISC.2025.22

**Related Version** *Full Version*: `https://arxiv.org/abs/2411.11544` [15]

**Funding** The research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (24-1323-A0001). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the Ministry of Education, Singapore.

## 1 Introduction

Detecting small subgraphs in distributed networks has recently attracted significant research interest [6, 9, 12, 16, 17, 20, 22, 25, 26, 27, 33, 30, 36]. Distributed subgraph finding plays an important role in understanding the *bandwidth limitation* in distributed networks: It is a classical problem where a straightforward $O(1)$-round solution exists with unlimited bandwidth, but becomes significantly more complex when bandwidth constraints are imposed.



Previous works on distributed subgraph findings mostly assumed a model in which the underlying network is *static*. However, distributed systems in real life may undergo topological changes over time: A node might crash, and a new connection might be formed between two existing nodes. To address this gap, Bonne and Censor-Hillel [6] initiated the study of distributed subgraph finding in dynamic networks to better capture the real-world behavior of networks of limited bandwidth. For the case where the target subgraph is a clique, they determined the tight bandwidth complexity bounds in nearly all settings. Later, Liu [36] extended this study to dynamic graphs with batched updates and resolved an open question of Bonne and Censor-Hillel [6, Open question 4]. However, several open questions remain, and very little is known about finding subgraphs beyond cliques. In this paper, we build upon their works [6, 36] to consider the remaining open questions and explore other target subgraphs beyond cliques.

## 1.1 Models

We now formally describe the models considered in this paper, which were introduced by Bonne and Censor-Hillel [6]. A dynamic network $\mathcal{G}$ is a sequence of graphs $\mathcal{G} = \{G^0, G^1, \ldots\}$. The superscript notation should not be confused with graph powers. The initial graph $G^0$ represents the state of the network at some starting point. For each $i > 1$, the graph $G^i$ is either identical to its preceding graph $G^{i-1}$ or obtained from $G^{i-1}$ by a single topological change.

Each node $u$ in the network is equipped with a unique identifier $\mathrm{ID}(u)$, and it knows the list of identifiers of all its neighbors. The communication is synchronous. In each round of communication, each node can send to each of its neighbors a message of $B$ bits, where $B$ denotes the *bandwidth* of the network.

We assume that each node initially knows the *entire* topology of the initial graph $G^0$. In particular, the set of all identifiers is global knowledge, so we may assume that the range of the identifiers is exactly $[n]$, where $n$ is the number of nodes in the network.

**Topological changes.** We consider four types of topological changes: node insertions, node deletions, edge insertions, and edge deletions. In the case of a node insertion, the adversary may connect the new node to an arbitrary subset of the existing nodes. Each node $u$ can only *indirectly* deduce that a topological change has occurred by comparing its list of neighbors $N_{G^i}(u)$ in the current round $i$ and its list of neighbors $N_{G^{i-1}}(u)$ in the previous round $i-1$. At most one topological change can occur in each round. Suppose at some round $i$, node $u$ detects that exactly one new node $v$ appears in its neighborhood list, then from this information only, node $u$ cannot distinguish whether edge $\{u, v\}$ is added or node $v$ is added in round $i$, if we are in the model where both edge insertions and node insertions are allowed. Similarly, suppose node $u$ detects that exactly one node $v$ disappears in its list. In that case, $u$ cannot distinguish whether edge $\{u, v\}$ is deleted or node $v$ is deleted, if we are in the model where both edge deletions and node deletions are allowed.

**Algorithms.** An algorithm can be designed to handle only one type of topological change or any combination of them. Throughout this paper, we only consider *deterministic* algorithms. We say an algorithm $\mathcal{A}$ is an $r$-round algorithm if $\mathcal{A}$ works in the following setting.

- Each topological change is followed with at least $r-1$ *quiet rounds*. Specifically, if a topological change occurs in round $i$, then we must have $G^i = G^{i+1} = \cdots = G^{i+r-1}$. Rounds $i+1, \ldots, i+r-1$ are quiet in the sense that no topological changes occur in these rounds.
- The output w.r.t. $G^i$ must be computed correctly by round $i+r-1$.

A one-round algorithm is not called a zero-round algorithm because, within each round, topological change occurs before any communication takes place. This setup allows the nodes to have one round of communication between the topological change and deciding the output within a single round. For example, if an edge $\{u, v\}$ is inserted in a round, then $u$ and $v$ can immediately communicate with each other along this edge within the same round.

We emphasize that all our algorithms and lower bounds in this paper are *deterministic*, although many of our lower bounds also extend to the randomized setting, as shown in the full version [15] of the paper.

## 1.2 Problems

We consider the four types of distributed subgraph finding problems introduced by Bonne and Censor-Hillel [6].

**Membership Listing** For the membership-listing ($\textsc{MemList}(H)$) problem, each node $v$ lists all the copies of $H$ containing $v$. In other words, for each copy of $H$ and each node $u$ in $H$, node $u$ lists $H$.

**Membership Detection** For the membership-detection ($\textsc{MemDetect}(H)$) problem, each node $v$ decides whether $v$ belongs to at least one copy of $H$.

**Listing** For the listing ($\textsc{List}(H)$) problem, every appearance of $H$ is listed by at least one node in the network. In other words, for each copy of $H$, there exists some node $u$ that lists $H$.

**Detection** For the detection ($\textsc{Detect}(H)$) problem, the existence of any $H$ must be detected by at least one node. Specifically, if the network does not contain $H$ as a subgraph, then all nodes must output No. Otherwise, at least one node must output Yes.

For both membership-detection and detection, the output of each node is binary (Yes/No) only, with no requirement to report the actual target subgraph. For both membership-listing and listing, each node outputs a list of the target subgraphs, using the node identifiers in the network. For the listing problem, the node $u$ responsible for listing $H$ is not required to be in $H$, and it is allowed that each copy of $H$ is listed by multiple nodes.

In the literature, the problem of deciding whether a subgraph isomorphic to $H$ exists is often referred to as *$H$-freeness*, whereas *detection* sometimes denotes the task of outputting a copy of $H$. We emphasize that our use of detection differs from this convention: In both $\textsc{MemDetect}(H)$ and $\textsc{Detect}(H)$, outputting a copy of $H$ is not required.

**Bandwidth complexities.** The $r$-round *bandwidth complexity* of a problem is defined as the minimum bandwidth $B$ for which there exists an $r$-round algorithm solving that problem with bandwidth $B$. Fix a target subgraph $H$, a round number $r$, and some type of topological changes. Let $B_{\textsc{MemList}}$, $B_{\textsc{MemDetect}}$, $B_{\textsc{List}}$, and $B_{\textsc{Detect}}$ denote the $r$-round bandwidth complexities of $\textsc{MemList}(H)$, $\textsc{MemDetect}(H)$, $\textsc{List}(H)$, and $\textsc{Detect}(H)$, respectively. The following observations were made by Bonne and Censor-Hillel [6].

▶ **Observation 1.1** ([6]). *Given any target subgraph $H$ and any integer $r$,*

$$B_{\textsc{Detect}} \leq B_{\textsc{List}} \leq B_{\textsc{MemList}}$$

*under any type of topological change.*

▶ **Observation 1.2** ([6]). *Given any target subgraph $H$ and any integer $r$,*

$$B_{\textsc{Detect}} \leq B_{\textsc{MemDetect}} \leq B_{\textsc{MemList}}$$

*under any type of topological change.*

**Nontrivial target subgraphs.** In this paper, we only focus on *nontrivial* target subgraphs $H$, meaning that we implicitly assume that $H$ is connected and contains at least three nodes:

- If $H$ is not connected, then all four problems are trivially impossible to solve, as we allow the network to be disconnected.
- If $H$ is connected with exactly two nodes, then all four problems are trivially solvable with zero communication.

For example, when we say that the one-round bandwidth complexity of $\textsc{MemDetect}(K_s)$ under edge insertions is $\Omega(\log\log n)$, we implicitly assume that $s \geq 3$.

## 1.3 Our Contributions

While Bonne and Censor-Hillel [6] settled most of the bandwidth complexity bounds for clique finding, they left five open questions, one of which [6, Open question 4] was resolved by Liu [36]. In this paper, we investigate the remaining ones.

**Finding cliques under edge insertions.** In [6, Open question 1] and [6, Open question 3], Bonne and Censor-Hillel asked for the tight bound on the one-round bandwidth complexity of membership-detection for triangles and larger cliques under *edge insertions* only. For triangles, they obtained two upper bounds $O(\log n)$ and $O(\sqrt{\Delta \log n})$ [6]. For larger cliques, they obtained an upper bound $O(\sqrt{n})$ which works even for the membership-listing problem [6]. In this work, we show that these problems admit a bandwidth lower bound of $\Omega(\log\log n)$, which holds even in bounded-degree networks.

▶ **Theorem 1.3.** *For any constant $s \geq 3$, the one-round bandwidth complexity of $\textsc{MemDetect}(K_s)$ under edge insertions is $\Omega(\log\log n)$, even in bounded-degree dynamic networks.*

Prior to our work, no lower bound was known for this problem. Moreover, we complement our lower bound with a new $O(\log\log n)$-bandwidth triangle finding algorithm in bounded-degree networks, which is capable of solving the more challenging problem of membership-listing.

▶ **Theorem 1.4.** *There exists a one-round algorithm of $\textsc{MemList}(K_3)$ under edge insertions with bandwidth $O(\Delta^2 \log\log n)$, where $\Delta$ is the maximum degree of the dynamic network.*

Combining Theorem 1.3 and Theorem 1.4, we obtain a tight bound of membership-detection for triangles in bounded-degree dynamic networks.

▶ **Corollary 1.5.** *The one-round bandwidth complexity of $\textsc{MemDetect}(K_3)$ under edge insertions is $\Theta(\log\log n)$ in bounded-degree dynamic networks.*

**Finding cliques under edge insertions and node insertions.** The remaining two open questions of Bonne and Censor-Hillel [6] considered the model where two types of topological changes are allowed. Specifically, [6, Open question 2] and [6, Open question 5] asked for the tight bound on the one-round bandwidth complexity of the listing problem for triangles and larger cliques under both *edge insertions* and *node insertions*. Previously, these problems were known to have an upper bound of $O(\log n)$ [6]. In this work, we show that these problems admit a bandwidth lower bound of $\Omega(\log\log\log n)$, which applies to the easier problem of detection and holds in bounded-degree networks.

▶ **Theorem 1.6.** *For any constant $s \geq 3$, the one-round bandwidth complexity of* $\textsc{Detect}(K_s)$ *under both node insertions and edge insertions is* $\Omega(\log\log\log n)$*, even in bounded-degree dynamic networks.*

Same as Theorem 1.3, prior to our work, no lower bound was known for this problem. Interestingly, we are also able to match this lower bound with a new $O(\log\log\log n)$-bandwidth algorithm for listing triangles in bounded-degree networks.

▶ **Theorem 1.7.** *There exists a one-round algorithm of* $\textsc{List}(K_3)$ *under edge insertions and node insertions with bandwidth* $O(\Delta \log\log\log n)$*, where* $\Delta$ *is the maximum degree of the dynamic network.*

Combining Theorem 1.6 and Theorem 1.7, we obtain a tight bound for triangle detection in bounded-degree dynamic networks.

▶ **Corollary 1.8.** *The one-round bandwidth complexity of* $\textsc{Detect}(K_3)$ *under edge insertions and node insertions is* $\Theta(\log\log\log n)$ *in bounded-degree dynamic networks.*

**Beyond cliques.** It appears to be a challenging task to extend the current results beyond cliques. In the static setting, while the round complexity for $k$-clique listing has been settled [9, 12, 16, 17, 22, 30], far less is known about target subgraphs that are not cliques. In the dynamic setting, Bonne and Censor-Hillel [6] highlighted that cliques are *unique* in that they can be found trivially in *one round* if bandwidth is unrestricted.

In this work, we demonstrate that it is possible to achieve meaningful results beyond cliques, despite the inherent difficulties. For subgraph finding beyond cliques, we present a *complete characterization* of the bandwidth complexity for the membership-listing problem across all target subgraphs, all numbers of rounds, and all four types of topological changes: node insertions, node deletions, edge insertions, and edge deletions (Table 1). Moreover, we show partial characterizations for one-round membership-detection (Table 2) and listing (Table 3).

Our contribution lies in finding structures in the apparent chaos: We identify relevant graph classes (e.g., complete multipartite graphs) and parameters (e.g., node-edge versions of distance, radius, and diameter) in the area of dynamic distributed subgraph finding. For the cases where a full characterization has yet to be achieved, we identify remaining challenging open problems and outline future research directions. Addressing these challenges will likely require developing novel techniques.

## 1.4 Additional Related Work

König and Wattenhofer [32] conducted a systematic study of classical network problems under various types of topological changes, such as node or edge insertions and deletions. They said that a combination of a problem and a topological change is *locally fixable* if an existing solution can be repaired in $O(1)$ rounds following a topological change in the network. Several subsequent studies have investigated dynamic distributed algorithms for local problems, such as independent set [3, 13], matching [39], and coloring [37]. The dynamic-LOCAL model was formalized in [1]. Distributed algorithms for highly dynamic networks, where multiple topological changes can occur within a single communication round, were examined in [4, 10]. Global distributed problems, such as consensus and information dissemination, have also been studied in the dynamic setting [18, 21, 28, 31, 34, 35, 40]. Dynamic distributed algorithms are closely related to the concept of *self-stabilization* – a key notion in distributed computing – where a distributed network undergoes various changes and must rapidly return to a stable state after some quiet time [19].

There is a long line of research studying distributed subgraph finding in the CONGEST model. The first breakthrough in triangle detection was achieved by Izumi and Le Gall [30], who demonstrated that triangle detection and listing can be completed in $\tilde{O}(n^{2/3})$ and $\tilde{O}(n^{3/4})$ rounds, respectively. After a series of work [9, 12, 14, 16, 17, 22], it was shown that $\tilde{\Theta}(n^{1-2/k})$ is the tight bound for $k$-clique listing via the use of expander decomposition and routing. Distributed cycle findings have also been extensively studied [20, 22, 25, 33]. Property-testing variants of the distributed subgraph finding problem have been explored [7, 11, 24, 26, 27]. The subgraph finding problem has also been investigated in other computational models beyond distributed and parallel computing [2, 5, 23]. For more on the distributed subgraph finding problem, see the survey by Censor-Hillel [8].

The study of distributed subgraph finding is partially motivated by the fact that many algorithms can be significantly improved if the underlying network does not contain certain small subgraphs. For instance, Pettie and Su showed distributed coloring algorithms for triangle-free graphs using fewer colors and rounds [38]. Similarly, Hirvonen, Rybicki, Schmid, and Suomela developed an efficient distributed algorithm to find large cuts in triangle-free graphs [29].

## 1.5 Roadmap

In Section 2, we review essential graph terminology and parameters and demonstrate how certain graph parameters determine the minimum number of rounds required for certain subgraph-finding tasks. In Section 3, we present a technical overview of our proofs. In Section 4, we conclude the paper with some open questions. In Section A, we provide tables that summarize our results for subgraph finding beyond cliques.

# 2 Preliminaries

In this section, we present the basic definitions used in the paper. In Section 2.1, we review essential graph terminology and parameters. In Section 2.2, we discuss some basic properties of these graph parameters. In Section 2.3, we demonstrate how specific graph parameters determine the minimum number of rounds required for MemList($H$) and MemDetect($H$), independent of any bandwidth constraints.

## 2.1 Graph Terminology

Given a graph $H$, an edge $e = \{u, v\} \in E(H)$, and a node subset $S \subseteq V(H)$, we write $H[S]$ to denote the subgraph of $H$ induced by node set $S$, write $H - S$ to denote the subgraph of $H$ induced by node set $V(H) \setminus S$, and write $H - e$ to denote the subgraph of $H$ induced by edge set $E(H) \setminus \{e\}$.

For a graph $G = (V, E)$, we have the following definitions.

▶ **Definition 2.1** (Eccentricity, diameter, radius, and center)**.**

| | |
|---|---|
| *Eccentricity of $u$:* | $\mathrm{ecc}_G(u) = \max\{\mathrm{dist}_G(u, v) : v \in V\}$ |
| *Diameter of $G$:* | $\mathrm{diam}(G) = \max\{\mathrm{ecc}_G(u) : u \in V\}$ |
| *Radius of $G$:* | $\mathrm{rad}(G) = \min\{\mathrm{ecc}_G(u) : u \in V\}$ |
| *Center of $G$:* | $\mathrm{center}(G) = \{u \in V : \mathrm{ecc}_G(u) = \mathrm{rad}(G)\}$ |

In other words, the eccentricity of a node is the maximum of its distance to other nodes, the diameter of a graph is the maximum eccentricity of its nodes, the radius of a graph is the minimum eccentricity of its nodes, and the center of a graph is a node subset containing all nodes with eccentricity equal to the radius.

We consider the "node-edge" version of the definitions above.

▶ **Definition 2.2** (Node-edge distance). *For any $u \in V$ and any $e = \{v, w\} \in E$, the node-edge distance between them is defined as* $\mathrm{dist}_G(u, e) := 1 + \min\{\mathrm{dist}_G(u, w), \mathrm{dist}_G(u, v)\}$.

We overload the notation $\mathrm{dist}(\cdot, \cdot)$ since it is easy to distinguish $\mathrm{dist}(u, v)$ and $\mathrm{dist}(u, e)$.

▶ **Definition 2.3** (Node-edge eccentricity, diameter, radius, and center).

*Node-edge eccentricity of $u$:*
$$\widetilde{\mathrm{ecc}}_G(u) = \max\{\mathrm{dist}_G(u, e) : e \in E\}$$

*Node-edge diameter of $G$:*
$$\widetilde{\mathrm{diam}}(G) = \max\{\widetilde{\mathrm{ecc}}_G(u) : u \in V\} = \max_{u \in V} \max_{e \in E} \mathrm{dist}_G(u, e)$$

*Node-edge radius of $G$:*
$$\widetilde{\mathrm{rad}}(G) = \min\{\widetilde{\mathrm{ecc}}_G(u) : u \in V\} = \min_{u \in V} \max_{e \in E} \mathrm{dist}_G(u, e)$$

*Node-edge center of $G$:*
$$\widetilde{\mathrm{center}}(G) = \{u \in V : \widetilde{\mathrm{ecc}}_G(u) = \widetilde{\mathrm{rad}}(G)\} = \{u \in V : \max_{e \in E} \mathrm{dist}_G(u, e) = \widetilde{\mathrm{rad}}(G)\}$$

The node-edge distance definition is needed to capture the complexity bounds for various problems discussed in this paper. For example, in $C_4$, any node can detect an edge deletion within one round, whereas in $C_5$, the node opposite the deleted edge cannot detect the topological change in a single round. This is reflected by the fact that $\widetilde{\mathrm{rad}}(C_4) = 2 < 3 = \widetilde{\mathrm{rad}}(C_5)$, while the standard radius definition $\mathrm{rad}(C_4) = 2 = \mathrm{rad}(C_5)$ is insufficient to capture the difference.

## 2.2 Properties of Graph Parameters

These distance-based parameters and their node-edge versions are closely related. Specifically, we have the following observation.

▶ **Observation 2.4.** *For any connected graph $H$,* $\mathrm{diam}(H) \le \widetilde{\mathrm{diam}}(H) \le \mathrm{diam}(H) + 1$.

**Proof.** Since for any node $u$ and any edge $\{v, w\}$, $|\mathrm{dist}_H(u, w) - \mathrm{dist}_H(u, v)| \le 1$, the observation follows directly from the definition of diam and $\widetilde{\mathrm{diam}}$.

$$\begin{aligned}\mathrm{diam}(H) &= \max_{u \in V} \max_{v \in V} \mathrm{dist}_H(u, v) \\ &\le \max_{u \in V} \max_{\{v,w\} \in E} 1 + \min\{\mathrm{dist}_H(u, w), \mathrm{dist}_H(u, v)\} = \widetilde{\mathrm{diam}}(H) \\ &\le 1 + \max_{u \in V} \max_{v \in V} \mathrm{dist}_H(u, v) = \mathrm{diam}(H) + 1.\end{aligned}$$

◀

The class of *complete multipartite graphs* plays a crucial role in the study of the complexity landscape of dynamic distributed subgraph finding, as it can be characterized in terms of node-edge diameter or node-edge independence.

▶ **Definition 2.5** (Complete multipartite graphs). *A graph $G$ is complete $k$-partite if its node set $V(G)$ can be partitioned into $k$ independent sets $S_1, S_2, \ldots, S_k$ such that for any two nodes $u, v \in V(G)$, $\{u, v\} \in E(G)$ if and only if $u \in S_i$ and $v \in S_j$ for some $i \neq j$. For any $k$, a complete $k$-partite graph is called a complete multipartite graph.*

▶ **Definition 2.6** (Node-edge independence). *In a graph $G$, a node $w$ and an edge $\{u, v\}$ are independent if $w$ is adjacent to neither $u$ nor $v$ in $G$. A graph $G$ is node-edge independent if at least one of its node-edge pairs are independent.*

In other words, a graph $G$ is node-edge independent if and only if it contains a co-$P_3$ (complement of a path with three nodes) as an induced subgraph. See Figure 1.

▶ **Lemma 2.7.** *A graph $G$ has no induced $P_3$ if and only if its connected components are cliques.*

**Proof.** If every connected component of $G$ is a clique, then any three nodes selected cannot induce a $P_3$. Conversely, suppose a graph $G$ has no induced $P_3$ and there is a connected component that is not a clique, then there is a pair of disconnected nodes $u$ and $v$ in the same component. The two nodes $u$ and $v$ must be connected by some minimum-length path $P_{uv}$ whose length is at least 2, since they are in the same connected component. Any three consecutive nodes on the path induce a $P_3$, which is a contradiction. ◀

▶ **Lemma 2.8.** *A graph $G$ is complete multipartite if and only if it is* not *node-edge independent.*

**Proof.** Observe that a graph is complete multipartite if and only if its complement is a disjoint union of cliques. Hence Lemma 2.7 implies that a graph $G$ is *not* node-edge independent $\iff$ $G$ does not contain a co-$P_3$ as an induced subgraph $\iff$ complement of $G$ is $P_3$-free $\iff$ $G$ is complete multipartite. ◀

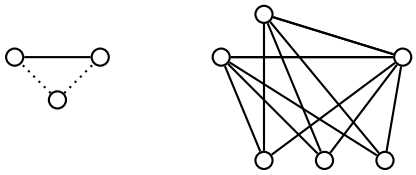

**Figure 1** Co-$P_3$ (left, dotted lines denote non-edges) and a complete multi-partite graph (right).

Now we characterize complete multipartite graphs in terms of node-edge diameter.

▶ **Observation 2.9.** *For any connected graph $H$ with at least three nodes, $\widetilde{\mathrm{diam}}(H) = 2$ if and only if $H$ is complete multipartite.*

**Proof.** If $\widetilde{\mathrm{diam}}(H) = 2$, then $H$ is not node-edge independent, since if node $w$ is independent of edge $e = \{u, v\}$, then $\mathrm{dist}_H(w, e) \geq 3$. Hence, $H$ is complete multipartite by Lemma 2.8. Conversely, suppose $H$ is complete multipartite, then it is not node-edge independent, and $\mathrm{dist}_H(w, e) \leq 2$ for all $w \in V(H)$ and $e \in E(H)$. Hence, $\widetilde{\mathrm{diam}}(H) \leq 2$. Since $H$ has at least three nodes, for any edge $e$, there is a node $w$ such that $w \notin e$. We have $\mathrm{dist}_H(w, e) \geq 2$ and $\widetilde{\mathrm{diam}}(H) \geq 2$. ◀

## 2.3 Locality Constraints

In this section, we investigate the minimum number of rounds required for the two problems MemList($H$) and MemDetect($H$) regardless of any bandwidth constraints, for any given target subgraph $H$. Due to the *local* nature of topological changes, the appearance or disappearance of a copy of $H$ might not be detectable for some nodes in $H$ in the same round or even after several rounds of communication.

See Figure 2 for an example. Suppose at some round $r$, edge $\{u, v\}$ is inserted, so the current graph is $C_5$. Assuming unlimited bandwidth, after one round of messaging, all the nodes highlighted can detect the appearance of $C_5$ due to the messages from $u$ or $v$. The only remaining node $w$ is unable to detect the appearance of $C_5$ after one round of messaging, as the information about the topological change has not reached $w$.


**Figure 2** Locality constraint for MemDetect($C_5$).

In the subsequent discussion, we define two graph parameters $r_H$ and $r'_H$, and use them to show the locality constraints for MemList($H$) and MemDetect($H$).

First, we define threshold $r_H$ and show that $r_H$ is a lower bound on the number of rounds needed for MemList($H$) or MemDetect($H$) under edge insertions or under edge deletions.

▶ **Definition 2.10** (Threshold $r_H$). *For any given graph $H$, define $r_H = \widetilde{\mathrm{diam}}(H) - 1$.*

▶ **Theorem 2.11** (Locality constraints, edge insertions/deletions). *Given any graph $H$, for any $T < r_H$, there exists no $T$-round algorithm for* MemList($H$) *or* MemDetect($H$) *under edge insertions or under edge deletions.*

**Proof.** Suppose $\mathrm{dist}_H(w, e) = \widetilde{\mathrm{diam}}(H)$, for a node $w \in V(H)$ and an edge $e = \{u, u'\} \in E(H)$. Consider the following dynamic graph $\mathcal{G}$ under edge insertions:

1. Initially, $G^0 = H - e$.
2. At round 1, consider two cases:
   a. The edge $\{u, u'\}$ is inserted. Thus $G^1 = H$.
   b. There is no change. Thus $G^1$ does not contain any subgraph isomorphic to $H$.

Since $T < r_H = \widetilde{\mathrm{diam}}(H) - 1 = \min\{\mathrm{dist}_H(w, u'), \mathrm{dist}_H(w, u)\}$, within $T$ rounds of communication, node $w$ must receive identical messages in both cases and cannot distinguish two cases, so any correct algorithm requires at least $r_H$ rounds.

A similar proof applies to the case of edge deletion. The only required modification is to start with $G^0 = H$ and then replace the insertion of $e$ with the deletion of $e$. ◀

Next, we define the threshold $r'_H$ and show that $r'_H$ is a lower bound on the number of rounds needed for MemList($H$) or MemDetect($H$) under node insertions or under node deletions.

▶ **Definition 2.12** (Threshold $r'_H$). *For any given graph $H$, define $r'_H = \mathrm{diam}(H) - 1$.*

▶ **Theorem 2.13** (Locality constraints, node insertions/deletions). *Given any graph $H$, for any $T < r'_H$, there exists no $T$-round algorithm for* MemList($H$) *or* MemDetect($H$) *under node insertions or under node deletions.*

**Proof.** Suppose $\mathrm{dist}_H(u, w) = \mathrm{diam}(H)$ for nodes $u, w \in V(H)$. For node insertions, we start with the graph $G^0 = H - \{u\}$, the subgraph of $H$ induced by $V(H) \setminus \{u\}$. Consider two cases:

1. Insert node $u$, along with all its incident edges in $H$. Thus $G^1 = H$.
2. No topological change.

Since $T < r'_H = \text{diam}(H) - 1 = \min\{\text{dist}_H(w, u') : u' \in N_H(u)\}$, node $w$ receives the same information within $T$ rounds in both cases. Therefore, for any $T$-round algorithm, $w$ is unable to correctly decide whether $H$ appears. We can use a similar design to prove the case of node deletions: Start with graph $H$, and then at round 1, either delete node $u$ or do nothing. Since $T < r'_H = \min\{\text{dist}_H(w, u') : u' \in N_H(u)\}$, the same analysis shows that node $w$ cannot decide whether $H$ disappears. ◀

We show a tighter bound in terms of $r_H$ for the case of node insertions. We remark that the same argument does work for node deletion since nodes are assumed to know the entire topology before any deletion and hence may decide based on one-sided information along the shortest path from $u$ to $x$ in the following example.

▶ **Theorem 2.14** (Locality constraints, node insertions). *Given any graph $H$, for any $T < r_H$, there exists no $T$-round algorithm for* MemList$(H)$ *or* MemDetect$(H)$ *under node insertions.*

**Proof.** According to Observation 2.4, we have either $r_H = \widetilde{\text{diam}}(H) - 1 = \text{diam}(H) - 1 = r'_H$ or $r_H = \widetilde{\text{diam}}(H) - 1 = \text{diam}(H) = r'_H + 1$. If $r_H = r'_H$, then the proof follows from Theorem 2.13. For the rest of the proof, we focus on the case where $r_H = \widetilde{\text{diam}}(H) - 1 = \text{diam}(H) = r'_H + 1$.

Suppose $\text{dist}_H(x, e) = \widetilde{\text{diam}}(H) = \text{diam}(H) + 1$ for node $x \in V(H)$ and edge $e = \{u, v\} \in E(H)$. We must have $\text{dist}_H(x, u) = \text{dist}_H(x, v) = \text{diam}(H)$. Otherwise, if $\text{dist}_H(x, u) \leq \text{diam}(H) - 1$, then $\text{dist}_H(x, e) \leq \text{diam}(H)$, contradicting our assumption. Now, consider the following dynamic graph $\mathcal{G}$.

1. Initially $G^0 = H - \{u\}$.
2. At round 1, consider two cases:
   a. Node $u$ is inserted with all its incident edges in $H$. Thus $G^1 = H$.
   b. Node $u$ is inserted with all its incident edges in $H$ excluding $\{u, v\}$. Thus $G^1 = H - e$.

For any $T < r_H = \text{diam}(H)$, node $x$ receives the same information within $T$ rounds in both cases. Therefore, for any $T$-round algorithm, $x$ cannot distinguish the two cases and cannot output correctly. ◀

**One-round solvability and complete multipartite graphs.** By Theorems 2.11 and 2.14 with $T = 1$, we know that $r_H \leq 1$, or equivalently $\widetilde{\text{diam}}(H) \leq 2$, characterizes the class of target subgraphs $H$ that permits one-round MemList$(H)$ and MemDetect$(H)$ algorithms for edge insertions, edge deletions, and node insertions. Observe that $\widetilde{\text{diam}}(H) = 1$ if and only if $H$ is a single edge, so all non-trivial target subgraphs $H$ satisfy $\widetilde{\text{diam}}(H) \geq 2$. Therefore, Observation 2.9 implies that, excluding trivial target subgraphs, complete multipartite graphs are exactly the class of graphs that permits one-round MemList$(H)$ and MemDetect$(H)$ algorithms for edge insertions, edge deletions, and node insertions.

# 3 Technical Overview

In this section, we present a technical overview of our proofs.

## 3.1 Lower Bounds for Finding Cliques

We start with the lower bounds for finding cliques.

**$\Omega(\log\log n)$ lower bound.** We present the core idea underlying the $\Omega(\log\log n)$ bandwidth lower bound for one-round MemDetect($K_s$) under edge insertions, as shown in Theorem 1.3. For simplicity and clarity, we focus on the case of membership-detecting triangles ($K_3$).

We start by describing the hard instance. Consider a set of $t = \log^{0.1} n$ nodes $\{x_1, x_2, \ldots, x_t\}$ and an independent set $I$. By connecting each $x_i$ to two distinct nodes from $I$, we form $t$ disjoint paths of length two, each resembling a triangle missing one edge. The graph is constructed by edge insertions over $2t$ rounds.

After constructing these paths, a single edge $e$ is inserted. Two scenarios are possible:

- Edge $e$ connects the two endpoints of one of the constructed paths, completing a triangle.
- Edge $e$ connects endpoints from two different paths, forming no triangle.

Let $(a, x_i, b)$ be one such path, and suppose $a$ is one endpoint of the newly inserted edge $e$. To correctly detect whether a triangle has been formed, node $a$ must determine whether the other endpoint of $e$ is node $b$. We show that if the bandwidth of the algorithm is $B = o(\log\log n)$, then no mechanism exists for $a$ to reliably make this distinction.

- Before the insertion of $e$, node $a$ might try to learn ID($b$) through $x_i$ during the initial $2t$ rounds. However, due to bandwidth limitations, $a$ can receive only $O(Bt) = o(\log n)$ bits, which is far too little to uniquely identify $b$ from the space of possible identifiers $[n]$.
- After the insertion of $e$, the two endpoints of $e$ can exchange $B$-bit messages. However, distinguishing whether they are part of the same path among $t$ candidates requires $\Omega(\log t)$ bits, while $B = o(\log\log n) = o(\log t)$ is again insufficient.

**$\Omega(\log\log\log n)$ lower bound.** We now turn to the proof of Theorem 1.6, which establishes that the one-round bandwidth complexity of Detect($K_s$) under both node insertions and edge insertions is $\Omega(\log\log\log n)$. For clarity, we again focus on the case of membership-detecting triangles.

The key reason this lower bound is exponentially smaller than the previous one is that Detect($K_s$) is inherently a much simpler problem than MemDetect($K_s$). In particular, if *only edge insertions are allowed*, then triangle detection is solvable with bandwidth $B = O(1)$: When an edge is inserted, its endpoints can simply broadcast a signal to their neighbors, and any node receiving two such signals can locally infer the existence of a triangle.

However, this strategy breaks down when *node insertions* are also permitted. Again, consider a path of the form $(a, x_i, b)$. Now, it is impossible to distinguish whether an edge $\{a, b\}$ has been inserted – completing a triangle – or whether a new node $c$ has been inserted with incident edges $\{a, c\}$ and $\{b, c\}$, where no triangle is created. The ambiguity arises because both scenarios lead to $x_i$ receiving signals from $a$ and $b$.

To formalize this, we use a construction similar to the one from the previous lower bound, but with a smaller parameter: $t = \log^{0.1}\log n$. We show that distinguishing the two scenarios requires $\Omega(\log\log\log n)$ bits of bandwidth.

Here is a brief sketch of the argument. We label each pair $(a, b)$, for $a, b \in I$, according to the messages transmitted across the edges $\{a, x_i\}$ and $\{b, x_i\}$ in the first $2t$ rounds, under the assumption that the path $(a, x_i, b)$ is formed. Since each message consists of $B$ bits and communication lasts $2t$ rounds, the total number of distinct labels is $s = 2^{O(Bt)}$. Since node insertion is allowed, nodes in $I$ without incident edges are considered as not yet inserted.

A Ramsey-type argument then implies the existence of a subset $\{a, b, c\} \subseteq I$ such that all of $(a, b)$, $(a, c)$, $(c, b)$ receive the same label. This means that, from the perspective of node $x_i$, the insertion of an edge $e = \{a, b\}$ versus the insertion of a node $c$ with two incident edges $\{a, c\}$ and $\{b, c\}$ becomes indistinguishable.

For the proof to work, the set $I$ must be sufficiently large relative to the number of distinct labels, which is $s = 2^{O(Bt)}$. This requirement is precisely why we set $t = \log^{0.1} \log n$ instead of the larger value $t = \log^{0.1} n$ used in the previous argument, resulting in a weaker lower bound of $\Omega(\log \log \log n)$. The full proof of Theorem 1.6 is more intricate, as it involves analyzing not only the perspective of each $x_i$, but also the views of the endpoints of the newly inserted edge $e$. For instance, the actual labeling in the proof requires quantifying over all $i \in [t]$, which increases the number of distinct labels to $2^{O(Bt^2)}$.

## 3.2 Upper Bounds for Finding Triangles

We now discuss our two triangle-finding algorithms, both of which achieve optimal bandwidth complexity in bounded-degree networks, matching our lower bounds.

Our algorithms build upon the one-round algorithm for MemDetect($K_3$) under edge insertions by Bonne and Censor-Hillel [6], which operates with bandwidth $B = O(\sqrt{\Delta \log n})$. Their approach uses two types of messages: one with $d$ bits, and another with $O\left(\frac{\Delta \log n}{d}\right)$ bits. The overall bandwidth complexity is optimized by choosing $d = \sqrt{\Delta \log n}$.

**$O(\log \log n)$ upper bound.** To prove Theorem 1.4, we extend the algorithm of Bonne and Censor-Hillel to design a one-round algorithm for MemList($K_3$) under edge insertions with a bandwidth complexity of $O(\Delta^2 \log \log n)$. In their original algorithm for MemDetect($K_3$), each $d$-bit message is a binary string where the $i$th bit indicates whether an incident edge was inserted $i$ rounds ago. We observe that this binary string can be compactly represented using $O(\Delta \log d)$ bits: For each recently inserted incident edge, use a number in $[d]$ to indicate its insertion time. Setting $d = \log n$ gives a bandwidth complexity of $O(\Delta \log \log n)$ for MemDetect($K_3$).

To handle the more demanding MemList($K_3$) problem, we introduce several modifications to the algorithm. Most notably, we must account not only for the edges incident to a node but also for those incident to its neighbors. This additional layer of information increases the size of the message by a factor of $\Delta$, resulting in a total bandwidth complexity of $O(\Delta^2 \log \log n)$.

**$O(\log \log \log n)$ upper bound.** We now turn to the proof of Theorem 1.7, where we design a one-round algorithm for List($K_3$) that handles both edge and node insertions with bandwidth $O(\Delta \log \log \log n)$. The exponential improvement stems from improving the size of the $O\left(\frac{\Delta \log n}{d}\right)$-bit message in the algorithm of Bonne and Censor-Hillel [6] to $O\left(\frac{\Delta \log \log n}{d}\right)$ bits. The purpose of this message is to transmit the list of neighborhood IDs, which contains $O(\Delta \log n)$ bits of information. As the transmission is done in $d$ rounds, the required message size is $O\left(\frac{\Delta \log n}{d}\right)$.

Our key idea lies in a new method for identifying triangles. Recall that a core challenge in our $\Omega(\log \log \log n)$ lower bound proof is to understand the inherent difficulty for a node $x$, with two non-adjacent neighbors $a$ and $b$, to distinguish between the insertion of an edge $\{a, b\}$ and the insertion of a node $c$ with two incident edges $\{a, c\}$ and $\{b, c\}$.

We develop a new algorithm to handle this instance. We let $x$ select an index $i$ such that the $i$th bits of ID($a$) and ID($b$) differ. Nodes $a$ and $b$ then report the $i$th bit of the identifier of their new neighbor. If an edge $\{a, b\}$ is inserted, then the reported bits will differ. If a node $c$, along with two incident edges $\{a, c\}$ and $\{b, c\}$, is inserted, then the bits will match. This comparison allows $x$ to distinguish between the two cases.

Sending an index requires only $O(\log \log n)$ bits, which is exponentially more efficient than sending the full identifier, which requires $O(\log n)$ bits. Consequently, the size of the $O\left(\frac{\Delta \log n}{d}\right)$-bit message in the algorithm of Bonne and Censor-Hillel [6] is reduced to $O\left(\frac{\Delta \log \log n}{d}\right)$ bits. Setting $d = \log \log n$ then yields the improved bandwidth complexity $O(\Delta \log \log \log n)$.

## 3.3 Membership-Listing

We obtain a *complete characterization* of the bandwidth complexity of MemList($H$) for every target subgraph $H$, every number of rounds $r$, and every type of topological change: node insertions, node deletions, edge insertions, and edge deletions. The full characterization is summarized in Table 1. In this overview, we omit node insertions and deletions, as they closely mirror the respective cases of edge insertions and deletions.

We begin with the case of *edge insertions*. For $r < r_H$, we have an impossibility result from Theorem 2.11. For $r \geq r_H$, the $r$-round bandwidth complexity depends on the structure of $H$:

- If $H$ is a complete multipartite graph that is not a clique, the complexity is $\Theta(n/r)$.
- If $H$ is not a complete multipartite graph, the complexity increases to $\Theta(n^2/r)$.

**Upper bounds.** The upper bounds follow from the observation that a node $v$ can list all subgraphs $H$ that contain it once it has an accurate view of its $r_H$-radius neighborhood. A brute-force approach would be to flood the entire graph topology using messages of size $O(n^2)$ after each topological change. Within $r_H$ rounds, this ensures all nodes acquire the required local view. If $r \geq r_H$, this communication can be spread over multiple rounds, reducing the bandwidth requirement to $O(n^2/r)$.

For the special case where $H$ is a complete multipartite graph, we have $r_H = 1$. This allows a more efficient approach: Each node simply broadcasts its list of neighbors as a binary string of length $n$, leading to a reduced bandwidth complexity of $O(n/r)$.

**Lower bounds.** We first discuss a general $\Omega(n/r)$ lower bound for any non-clique subgraph $H$. Let $\{u, v\}$ be a non-edge in $H$. Consider the graph resulting from replacing $u$ with an independent set $U'$, so each member of $U'$ corresponds to a copy of $H$. We then construct the graph via edge insertions, ensuring that the edges incident to $v$ are added last, leaving $v$ only $O(r)$ rounds to gather information about the graph. For $v$ to list all copies of $H$, it must learn the set $U'$, which requires $\Omega(n)$ bits of information, yielding a lower bound of $\Omega(n/r)$.

To strengthen the bound to $\Omega(n^2/r)$ when $H$ is not a complete multipartite graph, we use the fact that such a graph $H$ must contain an edge $e$ and a node $v$ such that neither endpoint of $e$ is adjacent to $v$. Now we apply a similar construction where $e$ is replaced with a bipartite graph, forcing $v$ to learn the bipartite graph in order to list all copies of $H$, which requires $\Omega(n^2)$ bits of information.

**Edge deletions.** We now consider the case of *edge deletions*. As with insertions, we have an impossibility result for $r < r_H$ from Theorem 2.11. For $r \geq r_H$, the $r$-round bandwidth complexity is $\Theta(n/r)$ whenever $H$ is not a clique.

A key difference between insertions and deletions is that edge insertions can merge disjoint components, potentially introducing many new subgraphs and requiring extensive information dissemination. In contrast, to handle deletions, it suffices to propagate the identifiers of the two endpoints of the deleted edge to a radius of $r_H$, which requires only $O(\log n)$ bits of information. This yields an upper bound of $O((\log n)/r)$.

To prove a matching lower bound, we reuse the construction from the $\Omega(n/r)$ lower bound. Suppose an edge deletion causes one of the $|U'|$ copies of $H$ to disappear. For a node $v$ to correctly identify *which* copy was affected, it must learn $\Omega(\log |U'|)$ bits. By letting $U'$ contain a polynomial number of nodes, this yields the desired $\Omega((\log n)/r)$ lower bound.

## 3.4 One-Round Membership-Detection

We now turn to the one-round bandwidth complexity of the MemDetect($H$) problem, for which we provide a partial characterization, see Table 2. Compared to MemList($H$), establishing lower bounds for MemDetect($H$) is more challenging, as it is harder to quantify the minimum information a node must obtain to detect the presence of a subgraph. On the algorithmic side, obtaining optimal upper bounds is also trickier: Since detection does not require listing the subgraph, there is greater flexibility in how the subgraph can be found. This flexibility enables a wider range of algorithmic techniques. In this overview, we focus on two representative results.

**Lower bound.** We show that the one-round bandwidth complexity of MemDetect($H$) under edge insertions is $\Omega(n)$ for any complete multipartite graph $H$ that is neither a star nor a clique. While the corresponding $\Omega(n/r)$ lower bound for MemList($H$) appears inherently tied to the listing requirement, we demonstrate that, with suitable modifications, the core idea can be adapted to the detection setting.

Since we cannot require a node $v$ to list all copies of $H$ in the constructed graph, we instead frame the argument in a preprocessing-plus-query model. We first build a graph that initially contains no copy of $H$, but is structured so that a copy can be formed in many different ways. The goal is to ensure that, in order for $v$ to detect the presence of $H$ following an edge insertion, it must have already learned a significant amount of information about the initial graph during the construction phase.

Specifically, we identify two non-adjacent nodes $u$ and $v$ in $H$ that share a common neighbor $w$. We construct a graph by removing the edge $\{u, w\}$ from $H$ and replacing node $u$ with an independent set $U'$. The graph is built via edge insertions, with the edges incident to $v$ and $w$ added at the end, leaving them only $O(1)$ rounds to learn about $U'$. We then insert a new edge $e$ incident to $w$, creating two possible scenarios: If the other endpoint of $e$ lies in $U'$, a copy of $H$ is formed; otherwise, it is not. For $v$ to decide correctly, $v$ and $w$ must learn the set $U'$ before the insertion of $e$, which requires $\Omega(n)$ bits of information.

**Upper bound.** As in the case of MemList($H$), membership-detection becomes significantly easier when the allowed topological change is a deletion rather than an insertion. In particular, we present a one-round $O(1)$-bandwidth algorithm for MemDetect($H$) that applies to any complete multipartite graph $H$, improving upon the $O(\log n)$ bound required for MemList($H$) under the same conditions.

For clarity, we describe our algorithm for the special case $H = C_4$, which captures the key ideas behind the more general algorithm that works for an arbitrary complete multipartite graph $H$. The core observation is that to detect a $C_4$, it suffices for each node $w$ to maintain, for every pair of its neighbors $u$ and $v$, the number of common neighbors they share, excluding $w$. Each such shared neighbor corresponds to a distinct copy of $C_4$ containing $w$, $u$, and $v$.

Maintaining these counters requires only one-bit messages. When a node detects that one of its neighbors has been deleted, it sends a one-bit signal to all its remaining neighbors. If a node $w$ receives such a signal from two neighbors $u$ and $v$ in the same round, it decrements the counter for the pair $\{u, v\}$ by one.

## 3.5 One-Round Listing

We study the one-round bandwidth complexity of the $\textsc{List}(H)$ problem for edge deletions and node deletions. See Table 3 for a summary of our results. In this overview, we focus on the case of edge deletions, as the case of node deletions is analogous.

We begin with the simpler cases. When $\widetilde{\text{rad}}(H) = 1$, $H$ is a star, so $\textsc{List}(H)$ is trivially solvable with zero bandwidth by allowing the star center to handle the listing. When $\widetilde{\text{rad}}(H) = 3$, $\textsc{List}(H)$ cannot be solved in one round, since there exists at least one edge $e$ in $H$ whose deletion cannot be communicated to all nodes within one round.

When $\text{rad}(H) = 1$, the graph $H$ has a center node $v$ adjacent to all other nodes, enabling a simple one-round one-bit algorithm: Whenever an edge is deleted, its endpoints send a signal to all their neighbors, allowing the center of $H$ to determine whether $H$ still exists.

We now turn to the remaining nontrivial case, where $\widetilde{\text{rad}}(H) = 2$ and $\text{rad}(H) = 2$. In this setting, we prove a tight $\Theta(\log n)$ bound. The upper bound is achieved by a simple protocol: When an edge $e = \{u, v\}$ is deleted, both $u$ and $v$ broadcast $\text{ID}(u)$ and $\text{ID}(v)$ to all their neighbors. This guarantees that if the deletion eliminates a copy of $H$, its center, who is responsible for listing the subgraph, can detect the change.

The lower bound is more involved and requires a novel construction. Let $V(H) = \{u_1, u_2, \ldots, u_m\}$. We replace each node $u_i$ in $H$ with an independent set of $n$ nodes $S_i = \{v_{i,1}, \ldots, v_{i,n}\}$. Moreover, we assume, based on the structure of $H$, that there exists a path of length two from $u_1$ to $u_m$ via $u_{m-1}$, but no direct edge between $u_1$ and $u_m$.

Using symmetry and the pigeonhole principle, we can assume that node $v_{1,1}$ must list at least $\Omega(n)$ distinct copies of $H$ of the form $\{v_{1,i}, v_{2,1}, v_{3,1}, \ldots, v_{m-1,1}, v_{m,j}\}$ for $i, j \in [n]$. Among these copies of $H$, there must be at least $\Omega(\sqrt{n})$ copies with distinct $i$-values, or at least $\Omega(\sqrt{n})$ with distinct $j$-values.

Assume the former holds, and let $I$ be the set of distinct $i$ values. Now consider deleting the edge $\{v_{1,i^*}, v_{m-1,1}\}$ for some $i^* \in I \setminus \{1\}$. Node $v_{1,1}$ must then stop listing all copies of $H$ that include this edge. Since there is no direct connection between $v_{1,1}$ and $v_{1,i^*}$, it must receive a message from $v_{m-1,1}$ that uniquely identifies $i^*$ among $\Omega(\sqrt{n})$ candidates, which requires $\Omega(\log n)$ bits. The argument for the latter case (distinct $j$-values) is similar.

# 4 Conclusions and Open Problems

In this work, we substantially extend the study of dynamic distributed subgraph finding initiated by Bonne and Censor-Hillel [6] in the *deterministic* setting. We establish *tight* one-round bandwidth bounds for triangle finding in bounded-degree dynamic networks: $\Theta(\log \log n)$ for membership-detection under edge insertions only (Corollary 1.5), and $\Theta(\log \log \log n)$ for detection when both edge and node insertions are allowed (Corollary 1.8). Before our work, no lower bound was known for these two problems. Moreover, we provide a *complete characterization* of the $r$-round bandwidth complexity of the membership-listing problem across all subgraphs and types of topological changes (Table 1). Despite these advances, many intriguing open problems remain.

**Beyond bounded-degree networks** While we obtain tight bounds for triangle finding in bounded-degree networks, the current upper and lower bounds remain unmatched for general unbounded-degree networks. Can stronger lower bounds be established for networks with higher degrees?

Specifically, for membership-detection, in Theorem 1.3, we establish a new lower bound of $\Omega(\log \log n)$ for the one-round bandwidth complexity of $\textsc{MemDetect}(K_s)$ under edge

insertions for any $s \geq 3$. While we provide a matching upper bound for MemList($K_3$) in *bounded-degree* networks in Theorem 1.4, this lower bound is not yet known to be tight for *unbounded-degree* networks, where the current best upper bound is $O(\log n)$ for $s = 3$ and $O(\sqrt{n})$ for $s \geq 4$ [6]. Closing these gaps remains an intriguing open question.

**Randomized algorithms** While we focus on deterministic algorithms in this paper, many of our lower bounds extend to randomized algorithms, as shown in the full version [15] of the paper. Yet the role of randomness in reducing bandwidth complexity for dynamic distributed subgraph finding is not well understood: Which problems exhibit an advantage for randomized over deterministic algorithms?

**Round-bandwidth tradeoffs** While our complete characterization of the membership-listing problem applies to $r$-round algorithms with an arbitrary round number $r$, the remainder of our results – and much of the existing literature – primarily focuses on the one-round scenario.

A particularly illustrative case is the membership-listing of cliques: In the one-round setting, the bandwidth complexity is $\Theta(\sqrt{n})$, whereas allowing two rounds reduces the bandwidth complexity to $\Theta(1)$ [6]. This stark contrast shows the potential benefits of additional communication rounds in lowering bandwidth requirements. Exploring how increased round numbers influence bandwidth complexity remains an interesting avenue for future research.

**Toward complete characterizations of the remaining problems** In this work, we provide partial characterizations for one-round membership-detection and listing. Can these characterizations be completed? What can be said about the detection problem? We discuss several specific open questions in the full version [15] of the paper.

## References


1 Amirreza Akbari, Navid Eslami, Henrik Lievonen, Darya Melnyk, Joona Särkijärvi, and Jukka Suomela. Locality in Online, Dynamic, Sequential, and Distributed Graph Algorithms. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2023.10.

2 Noga Alon, Tali Kaufman, Michael Krivelevich, and Dana Ron. Testing triangle-freeness in general graphs. *SIAM Journal on Discrete Mathematics*, 22(2):786–819, 2008. doi:10.1137/07067917X.

3 Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 815–826, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3188745.3188922.

4 Philipp Bamberger, Fabian Kuhn, and Yannic Maus. Local distributed algorithms in highly dynamic networks. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 33–42. IEEE, 2019. doi:10.1109/IPDPS.2019.00015.

5 Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*, pages 16–24, 2008. doi:10.1145/1839490.1839494.

6 Matthias Bonne and Keren Censor-Hillel. Distributed detection of cliques in dynamic networks. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.132.

7 Zvika Brakerski and Boaz Patt-Shamir. Distributed discovery of large near-cliques. In Idit Keidar, editor, *Proceedings of the 23rd International Symposium on Distributed Computing (DISC)*, volume 5805 of *Lecture Notes in Computer Science*, pages 206–220. Springer, 2009. doi:10.1007/978-3-642-04355-0_22.

8 Keren Censor-Hillel. Distributed Subgraph Finding: Progress and Challenges. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:14, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.3.

9 Keren Censor-Hillel, Yi-Jun Chang, François Le Gall, and Dean Leitersdorf. Tight distributed listing of cliques. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2878–2891, 2021. doi:10.1137/1.9781611976465.171.

10 Keren Censor-Hillel, Neta Dafni, Victor I. Kolobov, Ami Paz, and Gregory Schwartzman. Fast Deterministic Algorithms for Highly-Dynamic Networks. In Quentin Bramas, Rotem Oshman, and Paolo Romano, editors, *24th International Conference on Principles of Distributed Systems (OPODIS 2020)*, volume 184 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.OPODIS.2020.28.

11 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Computing*, 32(1):41–57, 2019. doi:10.1007/s00446-018-0324-8.

12 Keren Censor-Hillel, François Le Gall, and Dean Leitersdorf. On distributed listing of cliques. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 2020)*, 2020. doi:10.1145/3382734.3405742.

13 Keren Censor-Hillel, Elad Haramaty, and Zohar Karnin. Optimal dynamic distributed mis. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 217–226, 2016. doi:10.1145/2933057.2933083.

14 Keren Censor-Hillel, Dean Leitersdorf, and David Vulakh. Deterministic near-optimal distributed listing of cliques. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 271–280, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538434.

15 Yi-Jun Chang, Lyuting Chen, Yanyu Chen, Gopinath Mishra, and Mingyang Yang. The complexity landscape of dynamic distributed subgraph finding. *arXiv preprint arXiv:2411.11544*, 2024. doi:10.48550/arXiv.2411.11544.

16 Yi-Jun Chang, Shang-En Huang, and Hsin-Hao Su. Deterministic expander routing: Faster and more versatile. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 194–204, 2024. doi:10.1145/3662158.3662797.

17 Yi-Jun Chang, Seth Pettie, Thatchaphol Saranurak, and Hengjie Zhang. Near-optimal distributed triangle enumeration via expander decompositions. *Journal of the ACM*, 68(3):1–36, 2021. doi:10.1145/3446330.

18 Michael Dinitz, Jeremy T Fineman, Seth Gilbert, and Calvin Newport. Smoothed analysis of dynamic networks. *Distributed Computing*, 31:273–287, 2018. doi:10.1007/S00446-017-0300-8.

19 Shlomi Dolev. *Self-Stabilization*. MIT Press, 2000.

20 Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 2014)*, pages 367–376, 2014. doi:10.1145/2611462.2611493.

21 Chinmoy Dutta, Gopal Pandurangan, Rajmohan Rajaraman, Zhifeng Sun, and Emanuele Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 717–736. SIAM, 2013. doi:10.1137/1.9781611973105.52.

22 Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. Sublinear-time distributed algorithms for detecting small cliques and even cycles. In *Proceedings of the 33rd*

*International Symposium on Distributed Computing (DISC 2019)*, pages 15:1–15:16, 2019. doi:10.4230/LIPIcs.DISC.2019.15.

23 Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017. doi:10.1137/15M1054389.

24 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three notes on distributed property testing. In Andréa W. Richa, editor, *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, volume 91 of *LIPIcs*, pages 15:1–15:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.DISC.2017.15.

25 Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Proceedings of the 30th Symposium on Parallelism in Algorithms and Architectures (SPAA 2018)*, pages 153–162, 2018. doi:10.1145/3210377.3210401.

26 Pierre Fraigniaud and Dennis Olivetti. Distributed detection of cycles. *ACM Transactions on Parallel Computing (TOPC)*, 6(3):1–20, 2019. doi:10.1145/3322811.

27 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. Distributed testing of excluded subgraphs. In *International Symposium on Distributed Computing*, pages 342–356. Springer, 2016. doi:10.1007/978-3-662-53426-7_25.

28 Bernhard Haeupler and David Karger. Faster information dissemination in dynamic networks via network coding. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 381–390, 2011. doi:10.1145/1993806.1993885.

29 Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. Large cuts with local algorithms on triangle-free graphs. *The Electronic Journal of Combinatorics*, 24(4):4–21, 2017.

30 Taisuke Izumi and François Le Gall. Triangle finding and listing in CONGEST networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 2017)*, pages 381–389, 2017. doi:10.1145/3087801.3087811.

31 Irvan Jahja and Haifeng Yu. Sublinear algorithms in $T$-interval dynamic networks. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 317–327, 2020. doi:10.1145/3350755.3400228.

32 Michael König and Roger Wattenhofer. On local fixing. In Roberto Baldoni, Nicolas Nisse, and Maarten van Steen, editors, *Proceedings of the 17th International Conference on Principles of Distributed Systems (OPODIS), Nice, France*, volume 8304 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2013. doi:10.1007/978-3-319-03850-6_14.

33 Janne H. Korhonen and Joel Rybicki. Deterministic subgraph detection in broadcast CONGEST. In *Proceedings of the 21st International Conference on Principles of Distributed Systems (OPODIS 2017)*, pages 4:1–4:16, 2017. doi:10.4230/LIPIcs.OPODIS.2017.4.

34 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522, 2010. doi:10.1145/1806689.1806760.

35 Fabian Kuhn, Yoram Moses, and Rotem Oshman. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 1–10, 2011. doi:10.1145/1993806.1993808.

36 Quanquan C Liu. A note on improved results for one round distributed clique listing. *Information Processing Letters*, 181:106355, 2023. doi:10.1016/J.IPL.2022.106355.

37 Merav Parter, David Peleg, and Shay Solomon. Local-on-average distributed tasks. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 220–239. SIAM, 2016. doi:10.1137/1.9781611974331.CH17.

38 Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Information and Computation*, 243:263–280, 2015. doi:10.1016/J.IC.2014.12.018.

39 Shay Solomon. Fully dynamic maximal matching in constant update time. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 325–334. IEEE, 2016. doi:10.1109/FOCS.2016.43.

40 Haifeng Yu, Yuda Zhao, and Irvan Jahja. The cost of unknown diameter in dynamic networks. *Journal of the ACM (JACM)*, 65(5):1–34, 2018. doi:10.1145/3209665.

# A Tables

In the appendix, we provide tables that summarize our results for subgraph finding beyond cliques.

## A.1 Membership-Listing

We establish a complete characterization of the bandwidth complexity of $r$-round MemList($H$) for any target subgraph $H$, for any number of rounds $r$, under any single type of topological change. We emphasize that, while $H$ is assumed to be a constant-size graph, here we allow $r$ to be a function of $n$. See Table 1 for a summary of our results. Refer to Definitions 2.10 and 2.12 for the definition of $r_H$ and $r'_H$.

**Table 1** The bandwidth complexity of $r$-round MemList($H$).

| | $r \geq r_H$ | | | | $r < r_H$ |
|---|---|---|---|---|---|
| | Complete multipartite graphs ($r_H = 1$) | | | $r_H \geq 2$ | |
| | Cliques | | Others | | |
| | $r = 1$ | $r \geq 2$ | | | |
| **Edge insertions** | $\Theta(\sqrt{n})$ [6] | $\Theta(1)$ [6] | $\Theta(n/r)$ [New] | $\Theta(n^2/r)$ [New] | Impossible [2.11][2.14] |
| **Node insertions** | $\Theta(n/r)$ [6] | | $\Theta(n/r)$ [New] | $\Theta(n^2/r)$ [New] | |
| **Edge deletions** | $\Theta(1)$ [6] | | $\Theta((\log n)/r)$ [New] | | |

| | $r \geq r'_H$ | | $r < r'_H$ |
|---|---|---|---|
| | Cliques ($r'_H = 0$) | Others ($r'_H \geq 1$) | |
| **Node deletions** | 0 [6] | $\Theta((\log n)/r)$ [New] | Impossible [2.13] |

## A.2 One-Round Membership-Detection

We investigate the one-round bandwidth complexity of the MemDetect($H$) problem under any single type of topological change. A summary of our results is provided in Table 2. For the definitions of rad, diam, and $\widetilde{\text{diam}}$, see Definitions 2.1 and 2.3.

To see that the table for node deletions covers all cases, observe that diam $= 2$ implies rad $\in \{1, 2\}$ because rad $\leq$ diam, and from Observation 2.4 we infer that diam $= 2$ implies $\widetilde{\text{diam}} \in \{2, 3\}$. In the table for the remaining three types of topological changes, refer to the paragraph at the end of Section 2.3 for why complete multipartite graphs characterize one-round solvability.

For node deletions, our $O(1)$-bandwidth algorithm for the case where $\text{diam}(H) = \text{rad}(H) = \widetilde{\text{diam}}(H) = 2$ works for all complete multipartite graphs $H$. Recall from Observation 2.9 that a connected graph $H$ with at least three nodes is complete multipartite if

**Table 2** The bandwidth complexity of one-round MemDetect($H$).

| | Complete multipartite graphs | | | | Others |
|---|---|---|---|---|---|
| | Stars | $s$-cliques | | Others | |
| | | $s=3$ | $s\geq 3$ | | |
| **Edge insertions** | $\Theta(1)$ [New] | $O(\log n)$ [6] | $O(\sqrt{n})$ [6] | $\Omega(\log\log n)$ [1.3] | $\Theta(n)$ [New] | Impossible [2.9] |
| **Node insertions** | $\Theta(n)$ [6] | | | | $\Theta(n)$ [New] | [2.11] [2.14] |
| **Edge deletions** | $\Theta(1)$ [6] | | | | $O(\log n)$ [New] | |

| | diam $=1$ | diam $=2$ | | | diam $\geq 3$ |
|---|---|---|---|---|---|
| | | rad $=1$ | rad $=2$ | | |
| | | | $\widetilde{\text{diam}}=2$ | $\widetilde{\text{diam}}=3$ | |
| **Node deletions** | 0 [6] | $\Theta(1)$ [New] | $\Theta(1)$ [New] | $O(\log n)$ [New] | Impossible [New] |

and only if $\widetilde{\text{diam}}(H)=2$. For node deletions, the only case where we are unable to obtain a tight bound is when $\text{diam}(H)=\text{rad}(H)=2$ and $\widetilde{\text{diam}}(H)=3$. A notable example of such a graph $H$ is the 5-cycle $C_5$.

## A.3 One-Round Listing

We investigate the one-round bandwidth complexity of the problem of List($H$) for edge deletions and node deletions. For edge deletions, we obtain a complete characterization. For node deletions, we obtain an almost complete characterization, except for the case where $\text{rad}(H)=\text{diam}(H)=2$. See Table 3 for a summary of our results. Refer to Definitions 2.1 and 2.3 for the definition of rad, $\widetilde{\text{rad}}$, and diam.

**Table 3** The bandwidth complexity of one-round List($H$).

| | $\widetilde{\text{rad}}=1$ | $\widetilde{\text{rad}}=2$ | | $\widetilde{\text{rad}}\geq 3$ |
|---|---|---|---|---|
| | | rad $=1$ | rad $=2$ | |
| **Edge deletions** | 0 [New] | $\Theta(1)$ [New] | $\Theta(\log n)$ [New] | Impossible [New] |

| | rad $=1$ | rad $=2$ | | rad $\geq 3$ |
|---|---|---|---|---|
| | | diam $=2$ | diam $\geq 3$ | |
| **Node deletions** | 0 [New] | $O(\log n)$ [New] | $\Theta(\log n)$ [New] | Impossible [New] |