# Continuous Aggregation in Dynamic Ad-Hoc Networks*

Sebastian Abshoff and Friedhelm Meyer auf der Heide

Heinz Nixdorf Institute & Computer Science Department,
University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany
Phone: +49-5251-606433, Fax: +49-5251-606482
{abshoff,fmadh}@hni.upb.de

**Abstract.** We study a scenario in which $n$ nodes of a mobile ad-hoc network continuously collect data. Their task is to repeatedly update aggregated information about the data, e.g., the maximum, the sum, or the full information about all data received by all nodes at a given time step. This aggregated information has to be disseminated to all nodes.
We propose two performance measures for distributed algorithms for these tasks: The *delay* is the maximum time needed until the aggregated information about the data measured at some time is output at all nodes. We assume that a node can broadcast information proportional to a constant number of data items per round. A too large communication volume needed for producing an output can lead to the effect that the delay grows unboundedly over time. Thus, we have to cope with the restriction that outputs are computed not for all but only for a fraction of rounds. We refer to this fraction as the *output rate* of the algorithm.
Our main technical contributions are trade-offs between delay and output rate for aggregation problems under the assumption of $T$-stable dynamics in the mobile ad-hoc network: The network is always connected and is stable for time intervals of length $T \geq c \cdot \mathrm{MIS}(n)$ where $\mathrm{MIS}(n)$ is the time needed to compute a maximal independent set. For the maximum function, we are able to show that we can achieve an output rate of $\Omega(T/(n \cdot \mathrm{MIS}(n)))$ with delay $\mathcal{O}(n \cdot \mathrm{MIS}(n))$. For the sum, we show that it is possible to achieve an output rate of $\Omega(T^{5/2}/(n^2 \cdot \mathrm{MIS}(n)^3))$ with delay $\mathcal{O}(n^2 \cdot \mathrm{MIS}(n)^2/T^{3/2})$ if $T = \mathcal{O}(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$, and if $T = \Omega(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$, we can achieve an output rate of $\Omega(T/(n \cdot \mathrm{MIS}(n)^2))$ with delay $\mathcal{O}(n \cdot \mathrm{MIS}(n))$.

**Keywords:** Dynamic Networks, Aggregation, Token Dissemination

## 1 Introduction

There are various devices that communicate wirelessly with each other and observe their environment. For example, many smartphones are able to commu-

* This work was partially supported by the German Research Foundation (DFG) within the Priority Program "Algorithms for Big Data" (SPP 1736), by the EU within FET project MULTIPLEX under contract no. 317532, and the International Graduate School "Dynamic Intelligent Systems".

nicate with close-by smartphones via technologies such as Bluetooth, WiFi, or Near Field Communication. In addition, these smartphones are equipped with more and more sensors nowadays, e.g. accelerometers, magnetometers, gyroscopic, light, temperature, pressure, and humidity sensors to name only a few. In this paper, we consider a scenario where these devices have to form an ad-hoc network to process the huge amount of data collected by their sensors. The links in such a network are unstable and they change over time, and thus, the network is dynamic – especially, if the nodes are mobile. We are interested in providing all nodes of the network with aggregated information about their sensor data.

We model the ad-hoc network as a $T$-stable dynamic network, which is controlled by an adaptive adversary (as introduced by Haeupler and Karger [7]). This adversary is able to change all edges of the network every $T$ rounds, but is restricted to give a connected network. The set of nodes is fixed and nodes have unique IDs. A message of $\mathcal{O}(\log n)$ bits sent by some node in some round $r$ is delivered to all its neighbors in the graph of the following round $r + 1$.

In this adversarial model, we study two aggregation problems: the extremum problem (e.g., the maximum) and the summation problem (e.g., the sum). Here, the nodes are given inputs (e.g., integers) and they have to compute a function of all inputs of the network. While both problems can be solved with existing algorithms for dissemination problems, which allow for full reconstruction of each input, we show that they can be solved faster with algorithms that aggregate information within the process. For this purpose, we exploit certain properties of the binary operations used to define the problems. For speeding up the summation, we make use of the commutativity and associativity, and we exploit the additional idempotence of the extremum which makes the problem simpler.

Our main focus lies on continuous versions of these problems where nodes receive a new input in each round and have to compute a function of all inputs for a single round. Here, we refer to the *delay* of an algorithm as the maximum number of rounds between the round when inputs arrive at all nodes and the round the last node outputs the result of the function of these inputs. We are interested in algorithms that have a high *output rate*, i.e., algorithms that output as many results for different rounds at all nodes as possible. One way to continuously produce outputs is to start the execution of the non-continuous algorithm, output one result and restart the execution of the algorithm again to produce the next result. However, we show how to use a pipelining technique to increase this trivial output rate but only slightly increase the delay.

### 1.1 Our Contribution

In static networks, all three (non-continuous) problems can be solved in a linear number of rounds (cf. Section 4). The continuous variants of the extremum and summation problem can be solved with constant output rate, and the continuous dissemination problem with rate $\Omega\left(\frac{1}{n}\right)$ while all delays remain linear. Note that these results are asymptotically tight in static networks.

For 1-stable dynamic networks, we show that the (non-continuous) extremum problem still can be solved in a linear number of rounds (cf. Section 5). To

solve the other problems, we assume $T \geq c \cdot \text{MIS}(n)$ where $c$ is a sufficiently large constant and $\text{MIS}(n)$ is the number of rounds required to compute a maximal independent set in a graph with $n$ nodes (note that there are some restrictions under which this maximal independent set must be computed, cf. Section 5.1). Compared to the (non-continuous) dissemination problem, we are able to solve the (non-continuous) summation problem $\frac{T}{\text{MIS}(n)}$ times faster if $T = \mathcal{O}(\sqrt{n \cdot \text{MIS}(n)})$, and if $T = \Omega(\sqrt{n \cdot \text{MIS}(n)})$, this problem can be solved in a linear number of rounds. For the continuous extremum and summation problem, we prove non-trivial output rates, i.e., output rates that are higher than these obtained by executing the non-continuous algorithms over and over again. For the continuous extremum problem, we thereby increase the delay only slightly. If $T = \mathcal{O}(n^{2/3} \text{MIS}(n)^{2/3})$, we can achieve the same delay and a slightly smaller output rate for the continuous summation problem compared to the extremum problem. Besides these deterministic results (cf. Table 1), we show in the corresponding sections how randomization helps to improve these results.

Table 1: Overview about the deterministic results shown in this paper.

(a) Static Networks.

| | **Extremum** | **Summation** | **Dissemination** |
|---|---|---|---|
| **non-continuous** | | | |
| Running Time: | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ |
| **continuous** | | | |
| Delay: | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ |
| Output Rate: | $\Omega(1)$ | $\Omega(1)$ | $\Omega\left(\frac{1}{n}\right)$ |

(b) $T$-Stable Dynamic Networks with $T \geq c \cdot \text{MIS}(n)$.

| | **Extremum** | **Summation** | **Dissemination** |
|---|---|---|---|
| **non-continuous** | | | |
| Running Time: | | | |
| if $T = \mathcal{O}(\sqrt{n \cdot \text{MIS}(n)})$ | $\mathcal{O}(n)$ | $\mathcal{O}\left(\frac{n^2 \cdot \text{MIS}(n)}{T^2}\right)$ | $\mathcal{O}\left(\frac{n^2}{T}\right)$ |
| if $T = \Omega(\sqrt{n \cdot \text{MIS}(n)})$ | | $\mathcal{O}(n)$ | |
| **continuous** | | | |
| Delay: | | | |
| if $T = \mathcal{O}(n^{2/3} \text{MIS}(n)^{2/3})$ | $\mathcal{O}(n \cdot \text{MIS}(n))$ | $\mathcal{O}\left(\frac{n^2 \cdot \text{MIS}(n)^2}{T^{3/2}}\right)$ | $\mathcal{O}\left(\frac{n^2}{T}\right)$ |
| if $T = \Omega(n^{2/3} \text{MIS}(n)^{2/3})$ | | $\mathcal{O}(n \cdot \text{MIS}(n))$ | |
| Output Rate: | | | |
| if $T = \mathcal{O}(n^{2/3} \text{MIS}(n)^{2/3})$ | $\Omega\left(\frac{T}{n \cdot \text{MIS}(n)}\right)$ | $\Omega\left(\frac{T^{5/2}}{n^2 \cdot \text{MIS}(n)^3}\right)$ | $\Omega(\frac{T}{n^2})$ |
| if $T = \Omega(n^{2/3} \text{MIS}(n)^{2/3})$ | | $\Omega\left(\frac{T}{n \cdot \text{MIS}(n)^2}\right)$ | |

## 2 Models & Problems

We adapt the dynamic network model from Haeupler and Karger [7]: A dynamic network is a dynamic graph $G_r$, which consists of a fixed set of $n$ nodes. Each node has a unique ID that can be encoded with $\mathcal{O}(\log n)$ bits. Time proceeds in discrete, synchronous rounds. An adaptive adversary chooses a set of undirected edges $E_r$ defining the graph $G_r = (V, E_r)$ for round $r$. This adversary is only restricted to choose a connected graph in each round. In each round $r$, each node can send a message of $\Theta(\log n)$ bits, which is delivered to all neighbors in the graph $G_{r+1}$ in the following round $r+1$. The computational power of each node is unbounded. A dynamic network is called $T$-stable if the adversary is restricted to change the network only once every $T$ rounds. We assume throughout the paper that $T \leq n$. Furthermore, we assume that the nodes know both values $T$ and $n$. Note that $T$-stability is a stronger assumption than $T$-interval connectivity that has been proposed by Kuhn et al. [12] because $T$-interval connectivity only requires a stable spanning subgraph for $T$ rounds.

We study the following three (non-continuous) aggregation problems.

*Problem 1 (Extremum).* Let $(S, +)$ be a commutative and idempotent semigroup and let the elements of $S$ be representable with $\mathcal{O}(\log n)$ bits. Each node $i$ in the network receives as input an element $x_i \in S$. Let $f$ be defined by $f(x_1, x_2, \ldots, x_n) := \sum_{i=1}^{n} x_i$. All nodes of the network have to output $f(x_1, x_2, \ldots, x_n)$.

*Problem 2 (Summation).* Let $(S, +)$ be a commutative semigroup and let the elements of $S$ be representable with $\mathcal{O}(\log n)$ bits. Each node $i$ in the network receives as input an element $x_i \in S$. Let $f$ be defined by $f(x_1, x_2, \ldots, x_n) := \sum_{i=1}^{n} x_i$. All nodes of the network have to output $f(x_1, x_2, \ldots, x_n)$.

*Problem 3 (Dissemination).* Let $S$ be a structure representable with $\mathcal{O}(\log n)$ bits. Each node $i$ in the network receives as input an element $x_i \in S$ called token. All nodes of the network have to output $x_1, x_2, \ldots, x_n$.

In addition to that, we solve continuous versions of these problems where $f$ has not to be computed only once but several times for different inputs.

*Problems 4/5/6 (Continuous Extremum/Summation/Dissemination).* Define $f$ and $(S, +)$ as in the corresponding extremum/summation/dissemination problem. In each round $r$, each node $i$ in the network receives as input an element $x_{i,r} \in S$. For a subset of rounds $R \subseteq \mathbb{N}$ (defined by the algorithm) and each $r' \in R$, all nodes have to output $f(x_{1,r'}, x_{2,r'}, \ldots, x_{n,r'})$.

For example, consider $S$ to be a subset of $\mathbb{N}$ of size polynomial in $n$. Then, computing the sum is a summation problem whereas computing the maximum or the minimum of all inputs is an extremum problem. The dissemination problem is also known as the all-to-all token dissemination problem [12].

Note that solving these problems for one round, in general, requires more than just one round. Although it is possible to produce the result for each round

$r \in \mathbb{N}$, it could take longer and longer: Let $T$ be the number of rounds required to produce one output. Then, it is possible to output the result of round $r$ in round $r \cdot T$ by running the algorithm for each round one by one. Since we intend to run our algorithms for a long time, this is not a feasible approach and we do not want the number of rounds to produce one output to depend on the round when the computation is started.

Instead, we would like our algorithms to drop some rounds and produce outputs without this dependence. Intuitively, a good algorithm that continuously gives results produces as many results as possible and requires few rounds per output. This is captured by the following two definitions.

**Definition 1 (Output Rate).** *The* output rate *of an algorithm is defined as*

$$\lim_{r \to \infty} \frac{\#\text{results up to round } r}{r}.$$

**Definition 2 (Delay).** *The* delay *of an algorithm is defined as the maximum number of rounds between the round when inputs arrive and the round the function of these inputs is output by all nodes.*

## 3 Related Work

For static networks, one knows that many problems such as computing the maximum, sum, parity, or majority can be solved in linear time in a graph by first computing a spanning tree (see, e.g., Awerbuch [3]). More specifically, if $D$ is the diameter of the graph, all these functions can be computed in $\mathcal{O}(D)$ rounds. Beyond that, more complicated problems have been studied, e.g., selection problems [11] or the problem of computing the mode (most frequent element) [10].

The dynamic network model was introduced by Kuhn et al. [12]. In contrast to the model we use, they assumed that the number of nodes in the network is not known beforehand. In their setting, they studied two problems, the token dissemination and the counting problem. In the token dissemination problem, each node receives as input a token that has to be disseminated to all nodes such that in the end all nodes know all tokens. In the counting problem, the nodes have to determine the exact number of nodes in the network. They solved both problems with so-called token forwarding algorithms that are only allowed to store and forward tokens. Especially, these algorithms are not allowed to annotate or combine tokens or to send any other message except the empty message. For $T$-interval connected dynamic networks, they gave a deterministic $\mathcal{O}(n(n+k)/T)$ algorithm. This algorithm can also be used to solve the counting problem in $\mathcal{O}(n^2/T)$ rounds. If $n$ is known, $k$ tokens can be disseminated in $\mathcal{O}(nk/T + n)$ rounds. On the negative side, they showed that a subclass of knowledge-based token-forwarding algorithms needs $\Omega(nk/T)$ rounds for solving the token dissemination problem. In addition to that, they showed that even a centralized deterministic token-forwarding algorithm needs $\Omega(n \log k)$ rounds.

In a subsequent paper, the lower bound by Kuhn et al. was improved by Dutta et al. [6]. They showed that any randomized (even centralized) token-forwarding

algorithm requires $\Omega(nk/\log n + n)$ rounds. Furthermore, they gave an algorithm that can solve the $k$-token dissemination problem in $\mathcal{O}((n+k)\log n \log k)$ rounds w.h.p. in the presence of a weakly-adaptive adversary. For the offline case, they developed two randomized and centralized algorithms where one gives an $\mathcal{O}(n, \min\{k \log n\})$ schedule w.h.p. and the other gives an $\mathcal{O}((n+k)\log n^2)$ schedule w.h.p. if each node is allowed to send one token along each edge in each round. Haeupler and Kuhn [8] proved lower bounds when each node is allowed to send $b \leq k$ tokens per round or when the nodes have to collect a $\delta$-fraction of the tokens only. Their results are applicable for $T$-interval connected dynamic networks and dynamic networks that are $c$-vertex connected in each round.

Abshoff et al. [1] adapted the model by Kuhn et al. and restricted the adversary in a geometric setting. Here, each node has a position in the Euclidean plane and is moved by the adversary with maximum velocity $v_{\max}$. The nodes are able to reach all nodes within distance $R > 1$ and the adversary must keep the unit disk graph w.r.t. radius 1 connected. The $k$-token dissemination problem can be solved in $\mathcal{O}(n \cdot k \cdot \min\{v_{\max}, R\} \cdot R^{-2})$ rounds. In a different paper, Abshoff et al. [2] established a relation between counting and token dissemination by showing that a special token dissemination problem is at most as hard as counting the number of nodes in a directed variant of dynamic networks.

Haeupler and Karger [7] applied network coding techniques to the domain of dynamic networks. Their algorithm solves the token dissemination problem in $\mathcal{O}(n^2/\log n)$ rounds w.h.p. when both the message size and token size have length $\Theta(\log n)$ bits. With $T$-stability, they achieve a $T^2$ speedup. In the deterministic case, they can solve $k$-token dissemination in $\mathcal{O}\left(\frac{1}{\sqrt{\log nT}} \cdot n \cdot \min\{k, \frac{n}{T}\} + n\right) \cdot 2^{\mathcal{O}(\sqrt{\log n})}$ rounds. In the randomized case, they can solve $k$-token dissemination in $\mathcal{O}\left(\min\{\frac{nk}{T^2} + T^2 n \log^2 n, \frac{nk \log n}{T^2} + Tn \log^2 n, \frac{n^2 \log n}{T^2} + n \log n\}\right)$ rounds.

Cornejo et al. [5] studied a different aggregation problem where tokens have to be gathered at a minimum number of nodes. On the one hand, they proved that there is no algorithm with a good competitive ratio compared to an optimal offline algorithm. On the other hand, under the assumption that every node interacts with at least a $p$-fraction of the nodes, they give an algorithm that aggregates the tokens to a logarithmic number of nodes with high probability.

Mosk-Aoyama and Shah [15] showed how so-called separable functions can be approximated with a gossiping algorithm based on exponential random variables. Their techniques can also be applied in dynamic networks as Kuhn et al. [12] showed for approximate counting.

A main building block of this paper is the construction of maximal independent sets (MIS). The distributed algorithm by Luby [14] computes an MIS in expected $\mathcal{O}(\log n)$ rounds. It can also be shown that the number of rounds is $\mathcal{O}(\log n)$ w.h.p. [9,4]. The best known distributed deterministic algorithm by Panconesi and Srinivasan [16] computes an MIS in $2^{\mathcal{O}(\sqrt{\log n})}$ rounds. In growth-bounded graphs, Schneider and Wattenhofer [17] showed how to deterministically create an MIS in $\mathcal{O}(\log^* n)$ communication rounds. This is asymptotically optimal as Linial [13] gave a corresponding $\Omega(\log^* n)$ lower bound.

## 4 Static Networks

For the sake of a simpler presentation of the algorithms for $T$-stable dynamic networks, we shortly discuss how the problems can be solved in static networks.

The extremum problem can be solved in $n-1$ rounds: Each node $i$ initially broadcasts its input $x_i$. In every other round up to round $n-1$, each node $i$ takes its incoming messages $m_1, \ldots, m_l$ and broadcasts $\sum_{j=1}^{l} m_j + x_i$. In round $n-1$, $\sum_{j=1}^{l} m_j + x_i = f(x_1, \ldots, x_n)$ since all inputs must be contained in this sum and multiplicities cancel out due to the idempotence of the semigroup.

To solve the summation problem, we can first find the node with the smallest ID in $n-1$ rounds (this is an extremum problem). Then, in further $n-1$ rounds, we can build up a shortest path tree rooted at the node with the smallest ID. Along this tree, starting from the leaves up to the root, we can sum up the inputs and finally broadcast the result back from the root to all elements. We thereby guarantee that each summand is only considered once.

Finally, to solve the dissemination problem, we can build up a tree as for the summation problem. Then, each node in the tree sends a token it has not yet sent upwards to the root of the tree in each round. After that the tokens are sent one after the other from the root to the leaves.

**Proposition 1.** *In static networks, the extremum, the summation, and the dissemination problem can be solved in $\mathcal{O}(n)$ rounds.*

We could have used the algorithm that solves the dissemination problem to solve the extremum and the summation problem. However, we chose the aforementioned algorithms since they are similar to those we will use to solve these problems in their continuous versions in dynamic networks.

To continuously solve both the extremum and the summation problem, we build up a tree as before and apply a pipelining technique. The leaves of the tree start sending their inputs from the first round upwards. In the next round, the leaves start sending their inputs from the second round upwards and so on until round $n$. The nodes with distance $l$ to the leaves within the tree in round $r$ sum up the incoming messages from the level below and add their input from round $r-l$ if $r-l>0$. Then, after $n$ rounds, the results for round 1 to $n$ are sent one after the other from the root to the leaves. This gives $n$ outputs in $\mathcal{O}(n)$ rounds. Since the best possible output rate is 1 and the delay cannot be better than the diameter of the network, we get the following result.

**Proposition 2.** *In static networks, the continuous extremum problem and the continuous summation problem can be solved with delay $\mathcal{O}(n)$ and output rate $\Omega(1)$. The delay and the output rate are asymptotically optimal.*

For the dissemination problem, we cannot achieve better delays and output rates than these we get if we just solve the non-continuous version over and over again. The delay is bounded by the diameter of the network. For the output rate, consider $|S| = n$, a line of $n$ nodes and the edge $e$ with $\lceil \frac{n}{2} \rceil$ nodes on the left an $\lfloor \frac{n}{2} \rfloor$ nodes on the right. If the output rate was $\omega(\frac{1}{n})$, then $\omega(\frac{r}{n})$ outputs

must have been computed up to round $r$. We know that at most $\mathcal{O}(r \cdot \log n)$ bits could have passed $e$ from the left to the right. These bits separate up to $n^{\mathcal{O}(r)}$ instances. However, there are $\binom{|S|}{n/2}^{\omega(r/n)} = n^{\omega(r)}$ possibilities to choose the tokens on the left side. Hence, at least one output must be wrong.

**Proposition 3.** *In static networks, the continuous dissemination problem can be solved with delay $\mathcal{O}(n)$ and output rate $\Omega\left(\frac{1}{n}\right)$. The delay and the output rate are asymptotically optimal.*

## 5 *T*-stable Dynamic Networks

We now show how to solve the problems in $T$-stable dynamic networks. First, we introduce a graph patching technique.

### 5.1 Graph Patching in *T*-stable Dynamic Networks

In this section, we show how a $T$-stable dynamic network can be partitioned into patches such that aggregation is possible. This partitioning will help speeding up the summation problem, the continuous extremum, and the continuous summation problem. The following patching idea is adapted from Haeupler and Karger [7].

**Definition 3 ($D$-Patch, $D$-Patching).** *A* $D$-patch *of a graph $G = (V, E)$ is a rooted tree in $G$ that spans at least $\frac{D}{2}$ nodes and has depth at most $\frac{D}{2}$. A* $D$-patching *of a graph is a set of $D$-patches such that the sets of the nodes of all $D$-patches give a disjoint partition of $V$.*

Such a $D$-patching of $G$ can be distributedly computed by

1. finding a set of nodes in $G$ that form a maximal independent set (MIS) in $G^D$, i.e., the $D^{\text{th}}$ power[1] of $G$,
2. computing breadth-first trees rooted in each node of the MIS, where each non-MIS node is assigned to its closest MIS node.

Existing distributed MIS algorithms can be adapted for this approach. Let $\text{MIS}(n)$ be the number of rounds necessary to compute an MIS in a graph with $n$ nodes. If an MIS algorithm running in $G^D$ is simulated in $G$, one needs to take care that one edge in $G^D$ corresponds to a path of length up to $D$ in $G$. Therefore, an MIS algorithm is slowed down by a factor of $D$. In addition to that, it is also important to consider the congestion in the nodes of $G$ caused by paths that overlap during simulation. If an MIS algorithm can be modified appropriately, a $D$-patching can be computed in $\mathcal{O}(\text{MIS}(n) \cdot D)$ rounds.

**Proposition 4.** *[7,14,9,4] A graph $G$ can be partitioned into $D$-patches of size $\Omega(D)$ in $\mathcal{O}(\log(n) \cdot D)$ rounds w.h.p. with Luby's randomized MIS algorithm.*

[1] $G^D = (V, E^D)$ with $E^D = \{\{u, v\} | \exists \text{path between } u, v \in V \text{ of length} \leq D \text{ in } G\}$.

**Proposition 5.** *[7,16] A graph $G$ can be partitioned into $D$-patches of size $\Omega(D)$ in $\mathcal{O}(2^{\mathcal{O}(\sqrt{\log n})} \cdot D)$ rounds with Panconesi and Srinivasan's deterministic MIS algorithm.*

We would like to add that a patching can be computed faster in growth-bounded graphs.

**Proposition 6.** *A growth-bounded graph $G$ can be partitioned into $D$-patches of size $\Omega(D)$ in $\mathcal{O}(\log^*(n) \cdot D)$ rounds with Schneider and Wattenhofer's deterministic MIS algorithm.*

*Proof.* The algorithm by Schneider and Wattenhofer [17] can be modified such that it can be executed in our setting: In each competition and whenever the states are updated, each competitor is interested in the competitor $u$ in its neighborhood that has the minimum result $r_u^{j-1}$ among all its neighbors. In addition to that, the nodes only need to know whether there exists a competitor, a ruler, or a dominator in their neighborhood. Therefore, each node only needs to flood the minimum result of a competitor, whether there exists a ruler, and whether there exists a dominator for $D$ rounds. □

### 5.2 Non-Continuous Extremum

Despite the presence of an adaptive adversary, the extremum problem can be solved without the need for a graph patching. This is a tight result since even in a static network the extremum problem cannot be solved faster.

**Theorem 1.** *In 1-stable dynamic networks, the extremum problem can be solved in $\mathcal{O}(n)$ rounds.*

*Proof.* The algorithm that solves the extremum problem in dynamic networks is the same as the algorithm used for static networks. Each node $i$ initially broadcasts its input $x_i$. In every other round up to round $n-1$, each node $i$ takes all its incoming messages $m_1, \ldots, m_l$ and broadcasts $\sum_{j=1}^{l} m_j + x_i$. In round $n-1$, the sum $\sum_{j=1}^{l} m_j + x_i$ is equal to $f(x_1, \ldots, x_n)$ because it contains all inputs (each node causally influenced each other node after $n-1$ rounds [12]) and multiplicities cancel out due to the idempotence of the semigroup. □

### 5.3 Non-Continuous Summation

**Theorem 2.** *In $T$-stable dynamic networks with $T \geq c \cdot \mathrm{MIS}(n)$ for a sufficiently large constant $c$, the summation problem can be solved in*

- $\mathcal{O}\left(\frac{n^2 \cdot \mathrm{MIS}(n)}{T^2}\right)$ *rounds if* $T = \mathcal{O}(\sqrt{n \cdot \mathrm{MIS}(n)})$ *and*
- $\mathcal{O}(n)$ *rounds if* $T = \Omega(\sqrt{n \cdot \mathrm{MIS}(n)})$.

*Proof.* Consider the following algorithm for which we choose $D = \Theta\left(\frac{T}{\mathrm{MIS}(n)}\right)$.

1. Compute a $D$-patching.
2. In each patch, compute the sum of all inputs of the nodes in the patch.
3. Disseminate all partial sums of the patches to all nodes and sum them up.

If $c$ is large enough and $D$ is chosen properly, then we can do the first and the second step in at most $T$ rounds. Since each patch has size at least $\frac{D}{2}$ nodes, we have at most $\frac{2n}{D} = \mathcal{O}\left(\frac{n \cdot \mathrm{MIS}(n)}{T}\right)$ partial sums left. To disseminate them in the third step, we can use the token dissemination algorithm by Kuhn et al. [12] for $T$-interval connected dynamic networks. Thus, we solve the summation problem in $\mathcal{O}\left(\frac{n^2 \cdot \mathrm{MIS}(n)}{T^2} + n\right)$ rounds. □

**Corollary 1.** *In $T$-stable dynamic networks with $T \geq 2^{c \cdot \sqrt{\log n}}$ for a sufficiently large constant $c$, the summation problem can be solved in*

- $\mathcal{O}\left(\frac{n^2}{T^2} \cdot 2^{c \cdot \sqrt{\log n}}\right)$ *rounds if* $T = \mathcal{O}\left(\sqrt{n} \cdot \sqrt{2}^{c \cdot \sqrt{\log n}}\right)$ *and*
- $\mathcal{O}(n)$ *rounds if* $T = \Omega\left(\sqrt{n} \cdot \sqrt{2}^{c \cdot \sqrt{\log n}}\right)$.

Randomization allows us to speed up this computation if we use Luby's algorithm to compute the patching and Haeupler and Karger's randomized network coding algorithm for dissemination.

**Theorem 3.** *Let $L$ be the number of rounds Luby's algorithm needs to compute a maximal independent set with high probability. Then, in $T$-stable dynamic networks with $T \geq L$, the summation problem can be solved within the number of rounds as listed in Table 2a.*

*Proof.* Let $D = \frac{\frac{1}{2}T}{L+1}$. Then, we need at most $D \cdot L \leq \frac{1}{2}T$ rounds to compute a $D$-patching and have further $D \leq \frac{1}{2}T$ rounds to sum up all values in each patch. Now, we can use the randomized network coding algorithm by Haeupler and Karger [7] for dissemination. It needs

$$\mathcal{O}\left(\min\{\frac{nk}{T^2} + T^2 n \log^2 n, \frac{nk \log n}{T^2} + Tn \log^2 n, \frac{n^2 \log n}{T^2} + n \log n\}\right)$$

rounds to disseminate $k$ tokens with high probability. For different ranges of $T$, we need the following number of rounds with high probability.

1. $\mathcal{O}\left(\frac{n^2 \log n}{T^3}\right)$ if $T = \mathcal{O}(n^{1/5} \log^{-1/5} n)$
2. $\mathcal{O}(T^2 n \log^2 n)$ if $\Omega(n^{1/5} \log^{-1/5} n) = T = \mathcal{O}(n^{1/5})$
3. $\mathcal{O}\left(\frac{n^2 \log^2 n}{T^3}\right)$ if $\Omega(n^{1/5}) = T = \mathcal{O}(n^{1/4})$
4. $\mathcal{O}(Tn \log^2 n)$ if $\Omega(n^{1/4}) = T = \mathcal{O}(n^{1/3} \log^{-1/3} n)$
5. $\mathcal{O}\left(\frac{n^2 \log n}{T^2}\right)$ if $\Omega(n^{1/3} \log^{-1/3} n) = T = \mathcal{O}(n^{1/2})$
6. $\mathcal{O}(n \log n)$ if $\Omega(n^{1/2}) = T$

Note that the number of rounds in the second and fourth range increase with $T$. However, a $T$-stable dynamic network is also $\frac{T}{l}$-stable for any $l > 1$. Therefore, we can replace $T$ by the lower bound of the range.

1. $\mathcal{O}\left(\frac{n^2 \log n}{T^3}\right)$ if $T = \mathcal{O}(n^{1/5} \log^{-1/5} n)$
2. $\mathcal{O}(n^{7/5} \log^{8/5} n)$ if $\Omega(n^{1/5} \log^{-1/5} n) = T = \mathcal{O}(n^{1/5} \log n^{2/15})$
3. $\mathcal{O}\left(\frac{n^2 \log^2 n}{T^3}\right)$ if $\Omega(n^{1/5} \log n^{2/15}) = T = \mathcal{O}(n^{1/4})$
4. $\mathcal{O}(n^{5/4} \log^2 n)$ if $\Omega(n^{1/4}) = T = \mathcal{O}(n^{3/8} \log^{-1/2} n)$
5. $\mathcal{O}\left(\frac{n^2 \log n}{T^2}\right)$ if $\Omega(n^{3/8} \log^{-1/2} n) = T = \mathcal{O}(n^{1/2})$
6. $\mathcal{O}(n \log n)$ if $\Omega(n^{1/2}) = T$

This gives the results for the non-continuous summation in Table 2a. □

Table 2: Summation in $T$-Stable Dynamic Networks with $T \geq L$.

(a) (Non-Continuous) Summation.

| **Running Time** | **Range for $T$** |
|---|---|
| $\mathcal{O}\left(\frac{n^2 \log n}{T^3}\right)$ w.h.p. | if $L \leq T = \mathcal{O}(n^{1/5} \log^{-1/5} n)$ |
| $\mathcal{O}(n^{7/5} \log^{8/5} n)$ w.h.p. | if $\Omega(n^{1/5} \log^{-1/5} n) = T = \mathcal{O}(n^{1/5} \log n^{2/15})$ |
| $\mathcal{O}\left(\frac{n^2 \log^2 n}{T^3}\right)$ w.h.p. | if $\Omega(n^{1/5} \log n^{2/15}) = T = \mathcal{O}(n^{1/4})$ |
| $\mathcal{O}(n^{5/4} \log^2 n)$ w.h.p. | if $\Omega(n^{1/4}) = T = \mathcal{O}(n^{3/8} \log^{-1/2} n)$ |
| $\mathcal{O}\left(\frac{n^2 \log n}{T^2}\right)$ w.h.p. | if $\Omega(n^{3/8} \log^{-1/2} n) = T = \mathcal{O}(n^{1/2})$ |
| $\mathcal{O}(n \log n)$ w.h.p. | if $\Omega(n^{1/2}) = T \leq n$ |

(b) Continuous Summation.

| **Delay** | **Output Rate** | **Range for $T$** |
|---|---|---|
| $\mathcal{O}\left(\frac{n^2}{T^2}\right)$ w.h.p. | $\Omega\left(\frac{T^3}{n^2}\right)$ w.h.p. | if $L \leq T = \mathcal{O}(n^{1/4} \log^{-1/2} n)$ |
| $\mathcal{O}(n^{3/2} \log n)$ w.h.p. | $\Omega\left(\frac{T}{n^{3/2} \log n}\right)$ w.h.p. | if $\Omega(n^{1/4} \log^{-1/2} n) = T = \mathcal{O}(n^{1/4})$ |
| $\mathcal{O}\left(\frac{n^2 \log n}{T^2}\right)$ w.h.p. | $\Omega\left(\frac{T^3}{n^2 \log n}\right)$ w.h.p. | if $\Omega(n^{1/4}) = T = \mathcal{O}(n^{1/2})$ |
| $\mathcal{O}(n \log n)$ w.h.p. | $\Omega\left(\frac{T}{n \log n}\right)$ w.h.p. | if $\Omega(n^{1/2}) = T \leq n$ |

### 5.4 Continuous Extremum

**Theorem 4.** *In $T$-stable dynamic networks with $T \geq c \cdot \mathrm{MIS}(n)$ for a sufficiently large constant $c$, the continuous extremum problem can be solved with delay $\mathcal{O}(n \cdot \mathrm{MIS}(n))$ and output rate $\Omega\left(\frac{T}{n \cdot \mathrm{MIS}(n)^2}\right)$.*

*Proof.* Consider the following algorithm for which we choose $D = \Theta\left(\frac{T}{\mathrm{MIS}(n)}\right)$.

1. Each node $i \in V$ initializes $y_{i,r,0}$ with $x_{i,r}$ for $r = 1, \ldots, D$.
2. For $j = 1, \ldots, \frac{2n}{D}$ phases of $T$ rounds do:
   (a) Compute a $D$-patching.
   (b) Each node $i$ in each patch $P$, computes $y_{i,r,j}$ as the sum of $y_{i',r,j-1}$ for all nodes $i'$ from $P$ and all adjacent patches of $P$ for $r = 1, \ldots, D$.
3. Each node $i \in V$ returns $y_{i,r,\frac{2n}{D}}$ for $r = 1, \ldots, D$.

If $c$ is large enough and $D$ is chosen properly, we can do a) and b) in a stable phase of $T$ rounds. Consider any input $x_{i,r}$. We say a patch $P$ knows $x_{i,r}$ iff $x_{i,r}$ is contained in any $y_{i',r,j}$ for $i' \in P$. If there is a patch $P$ that does not know $x_{i,r}$ at the beginning a phase, then there is a patch $P^*$ that does not know $x_{i,r}$ at the beginning of the phase but knows $x_{i,r}$ at the end of the phase. Thus, at least $\frac{D}{2}$ nodes learn about $x_{i,r}$ in each phase until all nodes know $x_{i,r}$. We can conclude that after $\frac{2n}{D}$ phases all inputs $x_{i,r}$ are contained in all $y_{i',r,\frac{2n}{D}}$. Therefore, after $\frac{2n}{D} \cdot T = \mathcal{O}(n \cdot \mathrm{MIS}(n))$ rounds, we have generated $D$ outputs which gives the claimed delay and the output rate. □

**Corollary 2.** *In $T$-stable dynamic networks with $T \geq 2^{c \cdot \sqrt{\log n}}$ for a sufficiently large constant $c$, the continuous extremum problem can be solved with delay $\mathcal{O}(n \cdot 2^{c \cdot \sqrt{\log n}})$ and output rate $\Omega\left(\frac{T}{n \cdot 2^{c \cdot \sqrt{\log n}}}\right)$.*

Again, randomization allows us to speed up this computation.

**Theorem 5.** *Let $L$ be the number of rounds Luby's algorithm needs to compute a maximal independent set with high probability. Then, in $T$-stable dynamic networks with $T \geq L$, the continuous extremum problem can be solved with high probability with output rate $\Omega\left(\frac{T}{n \log n}\right)$ and delay $\mathcal{O}(n \log n)$.*

*Proof.* Let $D = \frac{\frac{1}{9}T}{L+1}$. Then, we need at most $D \cdot L \leq \frac{1}{2}T$ rounds to compute a $D$-patching and have further $9D \leq \frac{1}{2}T$ rounds to do the computations in the patch as we do in the proof of Theorem 4. If we repeat this $\frac{n}{D}$ times, then, w.h.p., we still have at least $\frac{n}{D}$ valid $D$-patchings. Therefore, w.h.p., after $\frac{n}{D} \cdot T = \mathcal{O}(n \log n)$ rounds, we can generate $D$ outputs which gives the claimed delay and output rate. □

### 5.5 Continuous Summation

**Theorem 6.** *In $T$-stable dynamic networks with $T \geq c \cdot \mathrm{MIS}(n)$ for a sufficiently large constant $c$, the continuous summation problem can be solved with delay*

- $\mathcal{O}\left(\frac{n^2 \cdot \mathrm{MIS}(n)^2}{T^{3/2}}\right)$ *if* $T = \mathcal{O}(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$ *and*
- $\mathcal{O}(n \cdot \mathrm{MIS}(n))$ *if* $T = \Omega(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$

*and output rate*

- $\Omega\left(\frac{T^{5/2}}{n^2 \cdot \mathrm{MIS}(n)^3}\right)$ *if* $T = \mathcal{O}(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$ *and*
- $\Omega\left(\frac{T}{n \cdot \mathrm{MIS}(n)^2}\right)$ *if* $T = \Omega(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$.

*Proof.* Consider the following algorithm for which we choose $D = \Theta\left(\frac{T}{\mathrm{MIS}(n)}\right)$.

1. Compute a $D$-patching.
2. In each patch, compute $\frac{D}{2}$ sums of all inputs of the nodes in the patch of $\frac{D}{2}$ rounds.
3. Disseminate all partial sums of the patches to all nodes and sum them up.

If $c$ is large enough and $D$ is chosen properly, then we can do the first and the second step in at most $T$ rounds. Since each patch has size at least $\frac{D}{2}$, we have at most $n$ partial sums left. Now, we use the network coding algorithm by Haeupler and Karger [7]. This algorithm is able to disseminate $k \leq n$ tokens in $\mathcal{O}\left(\left(\frac{n \cdot \mathrm{MIS}(n)}{\sqrt{T}} \cdot \min\{k \cdot \sqrt{\log n}, \frac{n}{T}\} + n\right) \cdot \mathrm{MIS}(n)\right)$ rounds. Thus, we can disseminate all up to $n$ partial sums in $\mathcal{O}\left(\left(\frac{n^2 \cdot \mathrm{MIS}(n)}{T^{3/2}} + n\right) \cdot \mathrm{MIS}(n)\right)$ rounds. If $T = \mathcal{O}(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$, we thereby generate $\frac{D}{2}$ outputs in $\mathcal{O}\left(T + \frac{n^2 \cdot \mathrm{MIS}(n)^2}{T^{3/2}}\right)$ rounds and achieve an output rate of $\Omega\left(\frac{T^{5/2}}{n^2 \cdot \mathrm{MIS}(n)^3}\right)$. If $T = \Omega(n^{2/3} \cdot \mathrm{MIS}(n)^{2/3})$, we are able to generate $\frac{D}{2}$ outputs in $\mathcal{O}(n \cdot \mathrm{MIS}(n))$ rounds and achieve an output rate of $\Omega\left(\frac{T}{n \cdot \mathrm{MIS}(n)^2}\right)$. □

**Corollary 3.** *In $T$-stable dynamic networks with $T \geq 2^{c \cdot \sqrt{\log n}}$ for a sufficiently large constant $c$, the continuous summation problem can be solved with delay*

- $\mathcal{O}\left(\frac{n^2 \cdot 2^{2c \cdot \sqrt{\log n}}}{T^{3/2}}\right)$ *if* $T = \mathcal{O}\left(n^{2/3} \cdot 2^{c \cdot \frac{2}{3} \cdot \sqrt{\log n}}\right)$ *and*
- $\mathcal{O}\left(n \cdot 2^{c \cdot \sqrt{\log n}}\right)$ *if* $T = \Omega\left(n^{2/3} \cdot 2^{c \cdot \frac{2}{3} \cdot \sqrt{\log n}}\right)$

*and output rate*

- $\Omega\left(\frac{T^{5/2}}{n^2 \cdot 2^{3c \cdot \sqrt{\log n}}}\right)$ *if* $T = \mathcal{O}\left(n^{2/3} \cdot 2^{c \cdot \frac{2}{3} \cdot \sqrt{\log n}}\right)$ *and*
- $\Omega\left(\frac{T}{2^{c \cdot \sqrt{\log n}}}\right)$ *if* $T = \Omega\left(n^{2/3} \cdot 2^{c \cdot \frac{2}{3} \cdot \sqrt{\log n}}\right)$.

Again, we can use Luby's algorithm to compute the patching and Haeupler and Karger's randomized network coding algorithm for dissemination.

**Theorem 7.** *Let $L$ be the number of rounds Luby's algorithm needs to compute a maximal independent set with high probability. Then, in $T$-stable dynamic networks with $T \geq L$, the continuous summation problem can be solved with the output rates and delays as listed in Table 2b.*

*Proof.* Let $D = \frac{\frac{1}{2}T}{L+1}$. Then, we need at most $D \cdot L \leq \frac{1}{2}T$ rounds to compute a $D$-patching and have further $2D \leq \frac{1}{2}T$ rounds to do the computations in the patch as we do in the proof of Theorem 6. Now, we can use the randomized network coding algorithm by Haeupler and Karger [7] for dissemination. It needs

$$\mathcal{O}\left(\min\{\frac{nk}{T^2} + T^2 n \log^2 n, \frac{nk \log n}{T^2} + Tn \log^2 n, \frac{n^2 \log n}{T^2} + n \log n\}\right)$$

rounds to disseminate $k$ tokens with high probability. For different ranges of $T$ and $k = n$, we need the following number of rounds with high probability.

1. $\mathcal{O}\left(\frac{n^2}{T^2}\right)$ if $T = \mathcal{O}(n^{1/4} \log^{-1/2} n)$
2. $\mathcal{O}(T^2 n \log^2 n)$ if $\Omega(n^{1/4} \log^{-1/2} n) = T = \mathcal{O}(n^{1/4} \log^{-1/4} n)$
3. $\mathcal{O}\left(\frac{n^2 \log n}{T^2}\right)$ if $\Omega(n^{1/4} \log^{-1/4} n) = T = \mathcal{O}(n^{1/2})$
4. $\mathcal{O}(n \log n)$ if $\Omega(n^{1/2}) = T$

Note that the number of rounds in the second range increases with $T$. However, a $T$-stable dynamic network is also $\frac{T}{l}$-stable for any $l > 1$. Therefore, we can replace $T$ by the lower bound of the range.

1. $\mathcal{O}\left(\frac{n^2}{T^2}\right)$ if $T = \mathcal{O}(n^{1/4} \log^{-1/2} n)$
2. $\mathcal{O}(n^{3/2} \log n)$ if $\Omega(n^{1/4} \log^{-1/2} n) = T = \mathcal{O}(n^{1/4})$
3. $\mathcal{O}\left(\frac{n^2 \log n}{T^2}\right)$ if $\Omega(n^{1/4}) = T = \mathcal{O}(n^{1/2})$
4. $\mathcal{O}(n \log n)$ if $\Omega(n^{1/2}) = T$

This gives the results for the continuous summation in Table 2b. □

## 6 Geometric Dynamic Networks

In the geometric dynamic network model by Abshoff et al. [1], nodes have positions in the Euclidean plane and the adversary is allowed to move the nodes with maximum velocity $v_{\max}$. Furthermore, the adversary must keep the unit disk graph w.r.t. radius 1 connected in each round and the nodes are able to reach all nodes within communication range $R > 1$. This special class of dynamic networks is $\left\lfloor \frac{R-1}{2 \cdot v_{\max}} \right\rfloor + 1$-interval connected because a node within distance 1 can increase its distance by at most $2v_{\max}$. If in addition to that $R \geq 2$, then

the communication graph contains a spanning $\Theta(R)$-connected subgraph that is stable for $\Theta(R \cdot v_{\max}^{-1})$ rounds. If nodes know their positions (e.g., by using GPS) or if they at least have the ability to sense the distances to their neighbors, then they are able to determine the stable subgraphs and the algorithms presented in this paper can be applied. For the MIS computation, we can use the algorithm by Schneider and Wattenhofer [17] since the stable subgraphs are growth-bounded. This yields improved results for geometric dynamic networks with $\text{MIS}(n) = \mathcal{O}(\log^* n)$ and $T = \Theta(R \cdot v_{\max}^{-1})$.

## 7 Conclusion and Future Prospects

We showed that both extremum and summation problems can be solved faster than dissemination problems in $T$-stable dynamic networks by exploiting properties such as commutativity, associativity, and idempotence. Especially, the idempotence seems to make the extremum problem a lot simpler. Future work could focus on new problems that have different properties and allow for aggregation. It would also be interesting to see if similar techniques could be applied to other dynamic models such as $T$-interval stable dynamic networks where only a connected subgraph must be stable for $T$ rounds. Furthermore, we would like to investigate lower bounds for these problems. In case of the summation problem, this could lead to a non-trivial lower bound for the counting problem (if $n$ is not known beforehand) since the counting problem can be reduced to a summation problem where each node starts with a 1 as input.

## References

1. S. Abshoff, M. Benter, A. Cord-Landwehr, M. Malatyali, and F. Meyer auf der Heide. Token dissemination in geometric dynamic networks. In P. Flocchini, J. Gao, E. Kranakis, and F. Meyer auf der Heide, editors, *ALGOSENSORS*, volume 8243 of *Lecture Notes in Computer Science*, pages 22–34. Springer, 2013.
2. S. Abshoff, M. Benter, M. Malatyali, and F. Meyer auf der Heide. On two-party communication through dynamic networks. In R. Baldoni, N. Nisse, and M. van Steen, editors, *OPODIS*, volume 8304 of *Lecture Notes in Computer Science*, pages 11–22. Springer, 2013.
3. B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems (detailed summary). In A. V. Aho, editor, *STOC*, pages 230–240. ACM, 1987.
4. S. Chaudhuri and D. P. Dubhashi. Probabilistic recurrence relations revisited. *Theor. Comput. Sci.*, 181(1):45–56, 1997.
5. A. Cornejo, S. Gilbert, and C. C. Newport. Aggregation in dynamic networks. In D. Kowalski and A. Panconesi, editors, *PODC*, pages 195–204. ACM, 2012.
6. C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In S. Khanna, editor, *SODA*, pages 717–736. SIAM, 2013.
7. B. Haeupler and D. R. Karger. Faster information dissemination in dynamic networks via network coding. In C. Gavoille and P. Fraigniaud, editors, *PODC*, pages 381–390. ACM, 2011.

8. B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. In M. K. Aguilera, editor, *DISC*, volume 7611 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2012.
9. R. M. Karp. Probabilistic recurrence relations. *J. ACM*, 41(6):1136–1150, 1994.
10. F. Kuhn, T. Locher, and S. Schmid. Distributed computation of the mode. In R. A. Bazzi and B. Patt-Shamir, editors, *PODC*, pages 15–24. ACM, 2008.
11. F. Kuhn, T. Locher, and R. Wattenhofer. Tight bounds for distributed selection. In P. B. Gibbons and C. Scheideler, editors, *SPAA*, pages 145–153. ACM, 2007.
12. F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In L. J. Schulman, editor, *STOC*, pages 513–522. ACM, 2010.
13. N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.
14. M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.
15. D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In E. Ruppert and D. Malkhi, editors, *PODC*, pages 113–122. ACM, 2006.
16. A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In S. R. Kosaraju, M. Fellows, A. Wigderson, and J. A. Ellis, editors, *STOC*, pages 581–592. ACM, 1992.
17. J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In R. A. Bazzi and B. Patt-Shamir, editors, *PODC*, pages 35–44. ACM, 2008.