Polynomial Counting in Anonymous Dynamic Networks with Applications to Anonymous Dynamic Algebraic Computations

Dariusz R. Kowalski[1]
Computer Science Department, University of Liverpool, Liverpool, UK
D.Kowalski@liverpool.ac.uk

Miguel A. Mosteiro[2]
Computer Science Department, Pace University, New York, NY, USA
mmosteiro@pace.edu

Abstract

Starting with Michail, Chatzigiannakis, and Spirakis work [20], the problem of *Counting* the number of nodes in Anonymous Dynamic Networks has attracted a lot of attention. The problem is challenging because nodes are indistinguishable (they lack identifiers and execute the same program) and the topology may change arbitrarily from round to round of communication, as long as the network is connected in each round. The problem is central in distributed computing as the number of participants is frequently needed to make important decisions, such as termination, agreement, synchronization, and many others. A variety of algorithms built on top of *mass-distribution* techniques have been presented, analyzed, and also experimentally evaluated; some of them assumed additional knowledge of network characteristics, such as bounded degree or given upper bound on the network size. However, the question of whether Counting can be solved deterministically in sub-exponential time remained open. In this work, we answer this question positively by presenting Methodical Counting, which runs in polynomial time and requires no knowledge of network characteristics. Moreover, we also show how to extend Methodical Counting to compute the sum of input values and more complex functions without extra cost. Our analysis leverages previous work on random walks in evolving graphs, combined with carefully chosen alarms in the algorithm that control the process and its parameters. To the best of our knowledge, our Counting algorithm and its extensions to other algebraic and Boolean functions are the first that can be implemented in practice with worst-case guarantees.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

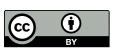
Keywords and phrases Anonymous Dynamic Networks, Counting, Boolean functions, distributed algorithms, deterministic algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.156

Related Version A full version of the paper is available at `https://arxiv.org/abs/1707.04282`.

Acknowledgements We thank Michal Koucký and Alessia Milani for useful discussions.

[1] Supported by Polish National Science Center grant UMO-2015/17/B/ST6/01897.
[2] Research work partially supported by Pace University Scholarly Research Award P1258 and the UK Royal Society Grant IES\R3\170293.



45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 156; pp. 156:1–156:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

How much information can one derive deterministically and distributedly from an arbitrarily evolving connected graph in polynomial time? Can we learn its size, or compute some simple Boolean functions, on its (distributed) input? In this work, we answer this question, posed in [20], in the affirmative. Specifically, we address first the problem of *Counting* the number of nodes of an Anonymous Dynamic Network (ADN) (and later extend to other algebraic functions) with a distributed and deterministic protocol. After a polynomial number of rounds of communication all nodes running our protocols obtain the result and stop. Our protocols resemble mass-distribution algorithms, however, our method is novel and quite complex as it has to deal with lack of node identities, lack of knowledge of network parameters, and adversarial topology changes.

The problem has been thoroughly studied [20, 10, 11, 12, 9, 21, 6] because Counting is central for distributed computing. Indeed, more complex tasks need the network size to make various decisions on state agreement, synchronization, termination, and others (e.g. [14, 15]). However, Anonymous Dynamic Networks pose a particularly challenging scenario. On one hand, nodes are indistinguishable from each other. For instance, they may lack identifiers or their number may be so massive that keeping record of them is not feasible. On the other hand, the topology of the network is highly dynamic. Indeed, the subsets of nodes that may communicate with each other may change all the time. All these features make ADN a valid model for anonymous ad hoc communication and computation.

In such a restrictive scenario, finding a way of providing theoretical guarantees of deterministic polynomial time has been elusive. Previous papers have either weaken the objective (e.g., computing only upper bound, only stochastic guarantees, etc.), assumed availability of network information (e.g., maximum number of neighbors, size upper bound, etc.), relied on a stronger model of communication, or provided only superpolynomial time guarantees.

Methodical Counting uses no information about the network. After completing its execution, all nodes obtain the exact size of the network and stop. Moreover, they stop all at the same time, allowing the algorithm to be concatenated with other computations.

Our algorithm is based on nodes continuously sharing some magnitude, which we call *potential*,[3] resembling *mass-distribution* and *push-pull* algorithms. Unlike previous algorithms, in Methodical Counting carefully and periodically (i.e. , "methodically") some potential is removed from the network, rather than greedily doing so continuously. This approach is combined with another methodological innovation testing whether the candidate value (for the network size) is within some polynomial range of the actual network size. This complex strategy yields an algorithm in which the progress in mass-distribution can be analyzed as a sequence of parametrized Markov chains (even though the algorithm itself is purely deterministic) enhanced by mass drift and alarms controlling the process and its parameters. Our analysis approach opens the path to study more complex tasks in Anonymous Dynamic Networks applying similar techniques.

Finally, we also present a variety of extensions of Methodical Counting to compute more complex functions. Most notably, we present an extension that, concurrently with finding the network size, computes the sum of input values held at each node without asymptotic time overhead. Having a method to compute the sum and network size, more complex computations are possible in polynomial time as well. Indeed, we also describe how to compute a variety of algebraic and Boolean functions.

[3] In previous related works this quantity, used in a different way, was termed *energy*. We steer away from such denomination to avoid confusion with node energy supply.

To the best of our knowledge, ours are the first algorithms for anonymous dynamic Counting and other algebraic computations that can be implemented in practice with worst-case polynomial-time guarantees.

Roadmap. The rest of the paper is organized as follows. Model and notation are detailed in Section 2. We overview previous work in Section 3 and present our results in Section 4. Section 5 includes the details of Methodical Counting, and we prove its correctness and running time in Section 6. Extensions to other functions are presented in Section 7.

2 Model, Problem, and Notation

The Counting Problem

The definition of the problem is simple. An algorithm solves the Counting Problem if, after completing its execution, all nodes have obtained the exact size of the network and stop.

Anonymous Dynamic Networks

The following model is customary in the Anonymous Dynamic Networks literature. We consider a network composed by a set V of $n > 1$ network *nodes* with processing and communication capabilities. It was shown in [20] that Counting cannot be solved in Anonymous Networks without the availability of at least one distinguished node in the network. Thus, we assume the presence of such node called *leader*. Aside from the leader, we assume that all other nodes are indistinguishable from each other. That is, we do not assume the availability of labels or identifiers, and all non-leader nodes execute exactly the same program.

Each pair of nodes that are able to communicate define a communication *link*, and the set of links is called the *topology* of the network. The nodes in a communication link are called *neighbors*. The event of sending a message to neighbors is called a *broadcast* or *transmission*. Nodes and links are reliable, in the sense that no communication or node failures occur. Hence, a broadcasted message is received by all neighbors. Moreover, links are *symmetric*, that is, if node a is able to send a message to node b, then b is able to send a message to a.

Without loss of generality, we discretize time in *rounds*. In any given round, a node may broadcast a message, receive all messages from broadcasting neighbors, and carry out some computations, in that order. Time needed for computations is assumed negligible.

The set of links among nodes may change from round to round, and nodes have no way of knowing which were the neighbors they had before. These topology changes are arbitrary, limited only to maintain the network connected in each round. That is, at any given round the topology is such that there is a *path*, i.e., a sequence of links, between each pair of nodes, but the set of links may change arbitrarily from round to round. This adversarial model of dynamics was called 1-*interval connectivity* in [19].

The following notation will be used. The maximum number of neighbors that any node may have at any given time, called the *dynamic maximum degree*, is denoted as Δ. An upper bound on Δ is denoted as $d_{\max}$. The maximum length of a shortest path between any pair of nodes at any given time is called the *dynamic diameter* and it is denoted as D. The maximum length of an opportunistic shortest path between any pair of nodes over many time slots is called the *chronopath* [13] and it is denoted as $\mathcal{D}$.

Table 1 Comparison of Counting protocols for Anonymous Dynamic Networks.

algorithm	needs		computes	stops?	complexity	
	size upper bound N	dynamic maximum degree u.b. $d_{\max}$			time	space
Degree Counting [20]		✓	$O(d_{\max}^n)$	✓	$O(n)$	
Conscious [10]	✓	✓	n	✓	$O(e^{N^2}N^3) \Rightarrow$ $O(e^{d_{\max}^{2n}} d_{\max}^{3n})$ using [20]	
Unconscious [10]			n	No	No theoretical bounds	
$\mathcal{A}_{\mathcal{O}^P}$ [11]		Oracle for each node	n	Eventually	Unknown	
EXT [9]			n	✓	$O(n^{n+4})$	EXPSPACE
Incremental Counting [21]		✓	n	✓	$O\left(n\,(2d_{\max})^{n+1}\,\frac{\ln n}{\ln d_{\max}}\right)$	
Methodical Counting [This work]			n	✓	$O(n^5 \ln^2 n)$	PSPACE

3 Previous Work

A comprehensive overview of work related to Anonymous Dynamic Networks can be found in a survey by Casteigts et al. [5] and references in the papers cited here. The directly related work overviewed in comparison with our results is summarized in Table 1.

With respect to lower bounds, it was proved in [8] that at least $\Omega(\log n)$ rounds are needed, even if D is constant. Also, $\Omega(\mathcal{D})$ is a lower bound since at least one node needs to hear about all other nodes to obtain the right count.

Counting and *Naming* was already studied in [20] for dynamic and static networks, showing that it is impossible to solve Counting without the presence of a distinguished node, even if the network is static. The Counting protocol requires knowledge of an upper bound on Δ, and obtains only an upper bound, which may be as bad as exponential.

Conscious Counting [10] computes the exact count, but it needs to start from an upper bound, and it takes exponential time only if the size upper bound is tight. In the same work and follow-up papers [11, 12], more challenging scenarios where Δ is unknown are studied, but protocols either do not terminate [10], or the protocol is terminated heuristically [12]. In experiments [12], such heuristic was found to perform well on dense topologies, but for other topologies the error rate was high. Another protocol in [11] is shown to terminate eventually, without running-time guarantees and under the assumption of having for each node an estimate of the number of neighbors in each round. In [20] it was conjectured that some knowledge of the network such as the latter would be necessary, but the conjecture was disproved later in [9]. On the other hand the protocol in [9] requires exponential space.

Incremental Counting, presented recently in [21], reduced exponentially the best-known running time guarantees. The protocol obtains the exact count, all nodes terminate simultaneously, the topology dynamics is only limited to 1-interval connectivity, it only requires polynomial space, and it only requires knowledge of of an upper bound ($d_{\max}$) on the dynamic maximum degree. The running time is still exponential, but reducing from doubly-exponential was an important step towards understanding the complexity of Counting.

In a follow-up paper [6], Incremental Counting was tested experimentally showing a promising polynomial behavior. The study was conducted on pessimistic inputs designed to slow the convergence, such as bounded-degree trees rooted at the leader uniformly chosen at random for each round, and a single path starting at the leader with non-leader nodes permuted uniformly at random for each round. The protocol was also tested on static versions of the inputs mentioned, classic random graphs, and networks where some disconnection is allowed. The results exposed important observations. Indeed, even for topologies that stretch the dynamic diameter, the running times obtained are below Δn^3. It was also observed that random graphs, as used in previous experimental studies [12], reduce the convergence time, and therefore are not a good choice to indicate worst-case behavior. These experiments showed good behavior even for networks that sometimes are disconnected, indicating that more relaxed models of dynamics, such as (α, β)-connectivity [13, 16], are worth to study. All in all, the experiments in [6] showed that Incremental Counting behaves well in a variety of pessimistic inputs, but not having a proof of what a worst-case input looks like, and being the experiments restricted to a range of values of n far from the expected massive size of an Anonymous Dynamic Network, a theoretical proof of polynomial time remained an open problem even from a practical perspective.

In a recent manuscript [2] a polynomial Counting algorithm is presented relying on the availability of an algorithm to compute average with polynomial convergence time. Such average computation is modeled as a Markov chain with underlying doubly-stochastic matrix, which requires topology information within two hops (cf. [23]). In our model of Anonymous Dynamic Network, such information is not available, and gathering it may not be possible due to possible topology changes from round to round.

Other studies also dealing with the time complexity of information gathering exist [7, 3, 24, 4, 22], but include in their model additional assumptions, such as the network having the same topology frequently enough.

4 Our Contributions

We present a deterministic distributed algorithm, which we call Methodical Counting, to compute the number of nodes in an Anonymous Dynamic Network. As opposed to previous works, our algorithm does not require any knowledge of network characteristics, such as dynamic maximum degree or an upper bound on the size. After $O(n^5 \ln^2 n)$ communication rounds of running Methodical Counting, all nodes obtain the network size and stop at the same round. To the best of our knowledge, this is the first polynomial deterministic Counting algorithm in the pure model of Anonymous Dynamic Network.

Our algorithm distributes potential in a mass-distribution fashion resembling previous works for Counting. The main novelty in our approach is that the leader participates in the process as any other node, removing potential only after it has accumulated enough. This approach allowed us to leverage previous work on random walks in evolving graphs. For this approach to work, we combine it with testing whether the candidate value for the network size is polynomially close to the actual value. Our approach also opens the path to study more complex computations in Anonymous Dynamic Networks using the same analysis.

Finally, we also present extensions of Methodical Counting to compute more complex functions. Most notably, we show how to modify Methodical Counting to compute the sum of input values held by nodes at the same time than counting. Having an algorithm to compute the network size and the sum of input values, we also show how to compute other algebraic and Boolean functions.

Algorithm 1 METHODICAL COUNTING algorithm for the leader. N is the set of neighbors of the leader in the current round. The parameters d, p, r and τ are as defined in Theorem 6.

```
1:  procedure COUNT
2:      ρ ← 0                                          // accumulator of consumed potential
3:      Φ ← 0                                          // current potential
4:      k ← 2                                          // current estimate
5:      status ← normal                                // status=normal|alarm|done
6:      while status ≠ done do                         // iterating epochs
7:          for phase = 1 to p do                      // iterating phases
8:              for round = 1 to r do                  // iterating rounds
9:                  Broadcast ⟨Φ, status⟩ and Receive ⟨Φ_i, status_i⟩, ∀i ∈ N
10:                 if status = normal and |N| ≤ d−1 and ∀i ∈ N : status_i = normal then
11:                     Φ ← Φ + Σ_{i∈N} Φ_i/d − |N|Φ/d     // update potential
12:                 else                                   // k is wrong
13:                     status ← alarm
14:                     Φ ← 1
                /* r rounds completed */
15:             if phase = 1 and Φ > τ then                // k is wrong
16:                 status ← alarm
17:                 Φ ← 1
18:             if status = normal then                    // prepare for next phase
19:                 ρ ← ρ + Φ
20:                 Φ ← 0
            /* p phases completed */
21:         if status = normal and k − 1 − 1/k ≤ ρ ≤ k − 1 then   // the size is k
22:             status ← done
23:         else                                           // prepare for next epoch
24:             ρ ← 0
25:             Φ ← 0
26:             k ← k + 1
27:             status ← normal
28:         for round = 1 to k do                          // disseminate termination
29:             Broadcast ⟨status⟩ and Receive ⟨status_i⟩, ∀i ∈ N
            /* epoch completed */
30:     return k
```

5 Methodical Counting

In this section we present METHODICAL COUNTING. First, we give the intuition of the algorithm, the details can be found in Algorithms 1 and 2. (References to algorithm lines are given as $\langle algorithm\#\rangle.\langle line\#\rangle$.)

Initially, the leader is assigned a potential of 0 and all the other nodes are assigned a potential of 1. Then, the algorithm is composed by epochs, each of which is divided into phases composed by rounds of communication. Epoch k corresponds to a size estimate k that is iteratively increased from epoch to epoch until the correct value n is found. Each epoch is divided into p phases. The purpose of each phase is for the leader to collect as much potential as possible from the other nodes in a mass-distribution fashion as follows.

Algorithm 2 METHODICAL COUNTING algorithm for each non-leader node i. N is the set of neighbors of i in the current round. The parameters d, p, r and τ are as defined in Theorem 6.

```
1:  procedure COUNT
2:      Φ ← 1                                                    // current potential
3:      k ← 2                                                    // current estimate
4:      status ← normal                                          // status=normal|alarm|done
5:      while status ≠ done do                                   // iterating epochs
6:          for phase = 1 to p do                                // iterating phases
7:              for round = 1 to r do                            // iterating rounds
8:                  Broadcast ⟨Φ, status⟩ and Receive ⟨Φ_i, status_i⟩, ∀i ∈ N
9:                  if status = normal and |N| ≤ d−1 and ∀i ∈ N : status_i = normal then
10:                     Φ ← Φ + Σ_{i∈N} Φ_i/d − |N|Φ/d           // update potential
11:                 else                                         // k is wrong
12:                     status ← alarm
13:                     Φ ← 1
               /* r rounds completed */
14:             if phase = 1 and Φ > τ then                      // k is wrong
15:                 status ← alarm
16:                 Φ ← 1
           /* p phases completed */
17:         for round = 1 to k do                                // disseminate termination
18:             Broadcast ⟨status⟩ and Receive ⟨status_i⟩, ∀i ∈ N
19:             if ∃i ∈ N : status_i = done then
20:                 status ← done
21:         if status ≠ done then
22:             k ← k + 1
23:             status ← normal
           /* epoch completed */
24:     return k
```

Each phase is composed by r rounds of communication. In each round, each node[4] broadcasts its potential and receives the potential of all its neighbors. Each node keeps only a fraction $1/d$ of the potentials received. The parameters p, r, and d are functions of k. The specific functions needed to guarantee correctness and saught efficiency are defined in Theorem 6. This varying way of distributing potential is different from previous approaches using mass distribution. After communication, each node updates its own potential accordingly (cf. Lines 1.11 and 2.10). That is, it adds a fraction $1/d$ of the potentials received, and subtracts a fraction $1/d$ of the potential broadcasted times the number of potentials received. Then, a new round starts.

At the end of each phase the leader "consumes" its potential. That is, it increases an internal accumulator ρ with its current potential, which is zeroed for starting the next phase (cf. Lines 1.19 and 1.20). A node stops the update of potential described, raises its potential to 1, and broadcasts an alarm in each round until the end of the epoch if any of the following happens: 1) at the end of the first phase its potential is above some threshold τ as defined in

[4] As opposed to previous work, in METHODICAL COUNTING the leader also follows this procedure.

Theorem 6 (cf. Lines 1.15 and 2.14), 2) at any round it receives more than $d-1$ messages (cf. Lines 1.12 and 2.11), or 3) at any round it receives an alarm (cf. Lines 1.12 and 2.11). The alarm for case 1) allows the leader to detect that the estimate is wrong when $k^{1+\epsilon} < n$ for some $\epsilon > 0$ (Lemmas 4 and 5), the alarm for case 2) allows the leader to detect that d is too small and hence the estimate is wrong, and the alarm for case 3) allows dissemination of all alarms. In the alarm status the potential is set to 1 to facilitate the analysis, but it is not strictly needed by the algorithm.

At the end of each epoch, the leader checks the value of ρ. If $k - 1 - 1/k \leq \rho \leq k - 1$ the current estimate is correct and the leader changes its status to "done" (cf. Line 1.21). Otherwise, all its variables are reset to start a new epoch with the next estimate (cf. Line 1.23). Before starting a new epoch the network is flooded with the status of the leader for k rounds (cf. Lines 1.28 and 2.17). If $k = n$, the leader initiates message "done" and the k rounds are enough for all the nodes to receive the "done" status and after completing the k rounds stop. Otherwise, nodes will not receive the "done" status and after completing the k rounds they start a new epoch.

6 Analysis

In this section we analyze Methodical Counting. References to algorithm lines are given as $\langle algorithm\# \rangle.\langle line\# \rangle$. We use standard notations $\mathbf{I}$ for the unit vector, and L_p for the norm of vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as $||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$, for any $p \geq 1$. Only for the analysis, nodes are labeled as $0, 1, 2, \ldots, n-1$, where the leader has label 0. The potential of a node i at the beginning of round s of phase t is denoted as $\Phi_{s,t}[i]$, the potential of all nodes is denoted as a vector $\mathbf{\Phi}_{s,t}$, and the aggregated potential is then $||\mathbf{\Phi}_{s,t}||_1$. The subindices s, t, or both are omitted sometimes for clarity. We will refer to the potential right after the last round of a phase as $\mathbf{\Phi}_{r+1}$. Round $r+1$ does not exist in the algorithm, but we use this notation to distinguish between the potential right before the leader consumes its own potential (cf. Line 1.23) and the potential at the beginning of the first round of the next phase.

First, we provide a broad description of our analysis of Methodical Counting. Consider the vector of potentials $\mathbf{\Phi}_i$ held by nodes at the beginning of any given phase i. The way that potentials are updated in each round (cf. Lines 1.11 and 2.10) is equivalent to the progression of a d-lazy random walk on the evolving graph underlying the network topology [1], where the initial vector of potentials is equivalent to an initial distribution $\mathbf{\Pi}_i$ on the overall potential $||\mathbf{\Phi}_i||_1$ and the probability of choosing a specific neighbor is $1/d$. For instance, the initial vector of potentials $\mathbf{\Phi}_0 = \langle 0, 1, 1, \ldots \rangle$, corresponds to a distribution $\mathbf{\Pi}_0 = \langle 0, 1/(n-1), 1/(n-1), \ldots \rangle$ on the initial $||\mathbf{\Phi}_0||_1 = n - 1$.

Note that our Methodical Counting is not a simple "derandomization" of the lazy random walk on evolving graphs. First, in the Anonymous Dynamic Network model neighbors cannot be distinguished, and even their number is unknown at transmission time (only at receiving time the node learns the number of its neighbors). Second, due to unknown network parameters, it may happen in an execution of Methodical Counting that the total potential received could be bigger than 1. Third, our algorithm does not know a priori when to terminate and provide result even with some reasonable accuracy, as the formulas on mixing and cover time of lazy random walks depend on (a priori unknown) number of nodes n. Nevertheless, we can still use some results obtained in the context of analogous lazy random walks in order to prove useful properties of parts of algorithm Methodical Counting, namely, some parts in which parameters are temporarily fixed and the number of received messages does not exceed parameter d.

It was shown in [1] that random walks on d-regular explorable evolving graphs have a uniform stationary distribution, and bounds on the mixing and cover time were proved as well. Moreover, it was observed that those properties hold even if the graph is not regular and d is only an upper bound on the degree.[5]

Thus, for the cases where d is an upper bound on the number of neighboring nodes, we analyze the evolution of potentials within each phase leveraging previous work on random walks on evolving graphs. Specifically, we use the following result which is an extension of Corollary 14 in [1].

▶ **Theorem 1.** *(Corollary 14 in [1].) After t rounds of a $d_{\max}$-lazy random walk on an evolving graph with n nodes, dynamic diameter D, upper bound on maximum degree $d_{\max}$, and initial distribution $\mathbf{\Pi}_0$, the following holds:*

$$\left|\left|\mathbf{\Pi}_t - \frac{\mathbf{I}}{n}\right|\right|_2^2 \leq \left(1 - \frac{1}{d_{\max} D n}\right)^t \left|\left|\mathbf{\Pi}_0 - \frac{\mathbf{I}}{n}\right|\right|_2^2 .$$

In between phases the leader "consumes" its potential, effectively changing the distribution at that point. Then, a new phase starts.

In Methodical Counting, given that d is a function of the estimate k, if the estimate is low, there may be inputs for which d is not an upper bound on the number of neighbors. We show in our analysis that in those cases the leader detects the error and after some time all nodes increase the estimate.

First, we prove correctness when $k = n$ in the following lemma. The proof, left to the full version of this paper, is based on upper bounding the potential left in the system after running the algorithm.

▶ **Lemma 2.** *If $d \geq k$ and $k = n$, after running the* Methodical Counting *protocol for $p \geq \frac{k}{1-1/k} \ln(k(k-1))$ phases, each of $r \geq 4dk^2 \ln k$ rounds, the potential ρ consumed by the leader is $k - 1 - 1/k \leq \rho \leq k - 1$.*

Lemma 2 shows that if $\rho > k-1$ or $\rho < k-1-1/k$ we know that the estimate k is wrong, but the complementary case, that is, $k-1-1/k \leq \rho \leq k-1$, may occur even if the estimate is $k < n$ and hence the error has to be detected by other means. To prove correctness in that case, we show first that if $k < n \leq k^{1+\epsilon}$ for some $\epsilon > 0$ the leader must consume $\rho > k-1$ potential if the protocol is run long enough. To ensure that $d \geq \Delta + 1$, we restrict $d \geq k^{1+\epsilon}$. The proof of the following lemma, based again on upper bounding the potential left in the system after running the algorithm, is left to the full version of this paper.

▶ **Lemma 3.** *If $1 < k < n \leq k^{1+\epsilon} \leq d$, $\epsilon > 0$, after running the* Methodical Counting *protocol for $p \geq \frac{(2+\epsilon)k^{1+\epsilon}}{1-1/k} \ln k$ phases, each of $r \geq (4+2\epsilon)dk^{2+2\epsilon} \ln k$ rounds, the potential ρ consumed by the leader is $\rho > k-1$.*

It remains to show that even if $n > k^{1+\epsilon}$ Methodical Counting still detects that the estimate is low. First, we prove the following two claims that establish properties of the potential during the execution of Methodical Counting. (Recall that we use round $r+1$ to refer to potentials at the end of the phase right before the leader consumes its potential in Line 1.23.) The proofs of both claims, based on observing the mass-distribution properties and the alarms in the algorithm, are left to the full version of this paper.

[5] Their analysis relies on Lemma 12, which bounds the eigenvalues of the transition matrix as long as it is stochastic, connected, symmetric, and non-zero entries lower bounded by $1/d$. Those conditions hold for all the transition matrices, even if the evolving graph is not regular.

▶ **Claim 1.** *Given an Anonymous Dynamic Network of n nodes running* Methodical Counting *with parameter d, for any round t of the first phase, such that $1 \leq t \leq r+1$, if d was larger than the number of neighbors of each node x for every round $t' < t$, then $||\mathbf{\Phi}_{1,t}||_1 = n-1$.*

▶ **Claim 2.** *Given an Anonymous Dynamic Network of n nodes running* Methodical Counting*, for any round t of any phase and any node x, it is $0 \leq \Phi_t[x] \leq 1$.*

It remains to show that even if $n > k^{1+\epsilon}$ Methodical Counting still detects that the estimate is low. We focus on the first phase. We define a threshold τ such that, after the phase is completed, all nodes that have potential above τ can send an alarm to the leader, as such potential indicates that the estimate is low. We show that the alarm must be received after $k^{1+\epsilon}$ further rounds of communication in the following lemma. The proof, based on bounding the "room" that nodes have up to the maximum potential, is left to the full version of this paper.

▶ **Lemma 4.** *For $\epsilon > 0$, after running the first phase of the* Methodical Counting *protocol, there are at most $k^{1+\epsilon}$ nodes that have potential at most $\tau = 1 - 1/k^{1+\epsilon}$.*

In our last lemma, we show that if $k^{1+\epsilon} < n$ the leader detects the error. The proof, based on bounding the number of nodes with low potential at the end of the first phase, is left to the full version of this paper.

▶ **Lemma 5.** *If $k^{1+\epsilon} < n$, $\epsilon > 0$, and $r \geq (4 + 2\epsilon - 2\ln(k^\epsilon - 1)/\ln k)dk^2 \ln k$, within the following $k^{1+\epsilon}$ rounds after the first phase of the* Methodical Counting *protocol, the leader has received an alarm message.*

Based on the above lemmata, we establish our main result in the following theorem.

▶ **Theorem 6.** *Given an Anonymous Dynamic Network with n nodes, after running* Methodical Counting *for each estimate $k = 2, 3, \ldots, n$ with parameters*

$$
\begin{aligned}
d &= k^{1+\epsilon}, \\
p &= \left\lceil \frac{(2+\epsilon)k^{1+\epsilon}}{1-1/k} \ln k \right\rceil, \\
r &= \left\lceil \left(4 + 2\epsilon + \max\left\{ 0, -\frac{2\ln(k^\epsilon - 1)}{\ln k} \right\} \right) dk^{2+2\epsilon} \ln k \right\rceil, \\
\tau &= 1 - 1/k^{1+\epsilon},
\end{aligned}
$$

where $\epsilon > 0$, all nodes stop after $\sum_{k=2}^{n}(pr + k)$ rounds of communication and output n.

Proof. Notice that the above parameters fulfill the conditions of the previous lemmas. First, we prove that Methodical Counting is correct. To do so, it is enough to show that for each estimate $k < n$ the algorithm detects the error and moves to the next estimate, and that if otherwise $k = n$ the algorithm stops and outputs k. We consider three cases: $k = n$, $k < n \leq k^{1+\epsilon}$, and $k^{1+\epsilon} < n$, for a chosen value of $\epsilon > 0$.

Assume first that $k < n \leq k^{1+\epsilon}$. Then, even if the leader does not receive an alarm during the execution, as shown in Lemma 3, at the end of the epoch in Line 1.21 the leader will detect that ρ is out of range and will not change its status to done. Therefore, no other node will receive a termination message (loop in Line 1.28), and all nodes will continue to the next epoch.

Assume now that $k^{1+\epsilon} < n$. Lemma 5 shows that within the following $k^{1+\epsilon}$ rounds after the first phase the leader has received an alarm message, even if no node has more than $d-1$ neighbors during the execution and alarms due to this are not triggered. For the given value of p and $k \geq 2$, the epoch has more than one phase. Therefore, within $k^{1+\epsilon}$ rounds into the second phase the leader will change to alarm status in Line 1.13, will not change its status to done later in this epoch, and no other node will receive a termination message. Hence, all nodes will continue to the next epoch.

Finally, if $k = n$, Lemma 2 shows that the accumulated potential ρ will be $k - 1 - 1/k \leq \rho \leq k - 1$. Thus, in Line 1.21 the leader will change its status to done, and in the loop of Line 1.28 will inform all other nodes that the current estimate is correct. The number of iterations of such loop are enough due to 1-interval connectivity.

The claimed running time can be obtained by inspection of the algorithm, either for the leader or non-leader since they are synchronized. Refer for instance to the leader algorithm in Algorithm 1. The outer loop in Line 1.6 corresponds to each epoch with estimates $k = 2, 3, \ldots, n$. For each epoch, Line 1.7 starts a loop of p phases followed by k rounds in Line 1.28. Each of the p phases has r rounds. Thus, the overal number of rounds is $\sum_{k=2}^{n}(pr + k)$. ◀

Choosing $\epsilon = \log_k 2$ and replacing in Theorem 6 yields the following corollary.

▶ **Corollary 7.** *The time complexity of* Methodical Counting *is* $O(n^5 \log^2 n)$.

7 Extensions

We argue that Methodical Counting can be extended to compute the sum of values stored in the nodes, and thus also the average (as it computes the number of nodes n), and other functions. Given that our Counting algorithm is based on mass-distribution, the standard approach could be to compute the average (by sharing the input values repeatedly until convergence to average) at the same time we compute the count. Then, the sum would be simply the average times the count. However, mass-distribution algorithms only converge to the result. That is, we may not get the exact sum with the procedure described. Then, a more careful method is needed.

Assume that each node of the Anonymous Dynamic Network initially stores a value, represented as a sequence of bits. Without loss of generality, we could assume that the value stored at the leader is zero; otherwise, the nodes could compute the sum of other initial values (with the leader value set up to 0), and later the leader could propagate its actual initial value appended to the message "done" at the end of the execution to be added to the computed sum of other nodes.

The modified Methodical Counting prepends the potential to the sequence. Instead of sending potential by the original Methodical Counting, each node transmits its current sequence (in which the potential stands in the first location). Changes at each position of the sequence are done independly by the same algorithm as used for the potential, cf. Algorithms 1 and 2. Re-setting the values, in the beginning of each epoch, means putting back the initial values of the sequence. It means that the modified algorithm maintains potential in exactly the same way as the original Methodical Counting, regardless of the initial values. At the end of some epoch, with number corresponding to the number of nodes n, all nodes terminate. When it happens, each node recalls the sequence stored in it at the end of the first phase of the epoch, multiplies the values stored at each position of the sequence by the epoch number n, and rounds each of the results to the closest integer;

then it sums up the subsequent values multiplied by corresponding (consecutive) powers of 2. Note that such "recalling" could be easily implemented by storing and maintaining the sequence after the first phase of each epoch.

We argue that the computed value is the sum of the initial values. It is enough to analyze how the modified algorithm processes values at one position of the sequence, as positions are treated independently; therefore, w.l.o.g. we assume that each node has value 0 or 1 in the beginning. Consider the last epoch before the leader sends the final sequence (in our case, representing one value). In the beginning of the epoch, the values are re-set to the original one, and manipulated independently according to the rules in Algorithms 1 and 2. Therefore, let us focus on the first phase of this epoch. Since we already proved that the estimate of the last epoch is equal to the number of nodes, the value of d in this epoch (and thus also in its first phase) is an upper bound on the node degree. Thus, the mass distribution scaled down by the sum of the initial values behaves exactly the same as the probabilities of being at nodes in the corresponding round of the lazy random walk, with parameter d and starting from initial distribution equal to the initial values divided by the sum. Since the length of the phase is set up to guarantee that the distribution is close to the stationary uniform within error $1/n$, and the sum of bits is not bigger than n, at the end of the phase the value stored by each node is close to the sum (i.e., scaling factor) divided by n by at most $1/n^4$ (cf. proof of Lemma 2). Therefore, after multiplying it by n, each node gets value of sum within error of at most $1/n^3$, which after rounding will give the integer equal to the value of the sum.

Once having the number n and the sum, each node can compute the average. As argued in [17], the capacity of computing the sum of the input values makes possible the computation of more complex functions. As opposed to [17] where the computation only converges, our approach outputs the exact sum. Therefore, the extension to database queries that can be approximated using *linear synopses*[6] is straightforward. Boolean functions $f : \{0,1\}^n \to \{0,1\}$, such as AND (sum $= n$), OR (sum > 0), and XOR (sum $= 1$), as well as their complementaries NAND (sum $\neq n$), NOR (sum $= 0$), and XNOR (sum $\neq 1$), can also be implemented having n and the sum. This applies also to other "symmetric" (i.e., do not depend on the order of variables) Boolean functions, as they could be computed based on computed sum of ones and n [18]. Maximum (L_∞ norm) and minimum can be computed subsequently by flooding. That is, each node broadcasts the maximum and minimum input values seen so far. Due to 1-interval connectivity within n rounds all nodes have the answers.

Note that all these computations, including the Methodical Counting, could be done using only polynomial estimates of values, that is, with messages of length $O(\log n)$, multiplied by the maximum number of coordinates of any of the initial values. This could be also traded for time: we could use only messages of length $O(\log n)$ with time increased by the maximum number of coordinates of any initial value (which is still polynomial in the size of the input,[7] which in this case is at least n plus the maximum number of coordinates).

8 Open Directions

Straightway questions emerging from our work include existence of polynomial (in n) lower bound and improvement of our upper bound. One of the potential ways could be through investigating bi-directional relationships between random processes and computing algebraic functions in Anonymous Dynamic Network. Extending the range of polynomially computable

[6] Additive functions on multisets, e.g. $f(A \cup B) = f(A) + f(B)$.
[7] The input in this case is distributed among the nodes, and each node possesses at least one bit

functions is another intriguing future direction. Finally, generalizing the model by not assuming connectivity in every round or dropping assumption on synchrony could introduce even more challenging aspects of communication and computation, including group communication and its impact on the common knowledge about the system parameters.

References

1. Chen Avin, Michal Kouckỳ, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Automata, languages and programming*, pages 121–132. Springer, 2008.
2. Roberto Baldoni and Giuseppe A Di Luna. Counting on anonymous dynamic networks: Bounds and algorithms. manuscript, 2016.
3. Siddhartha Banerjee, Aditya Gopalan, Abhik Kumar Das, and Sanjay Shakkottai. Epidemic spreading with external agents. *IEEE Transactions on Information Theory*, 60(7):4125–4138, 2014.
4. Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.
5. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.
6. Maitri Chakraborty, Alessia Milani, and Miguel A. Mosteiro. Counting in practical anonymous dynamic networks is polynomial. In *Proceedings of the 4th International Conference on Networked Systems*, volume 9944 of *Lecture Notes in Computer Science*, pages 131–136, 2016.
7. Yuxin Chen, Sanjay Shakkottai, and Jeffrey G Andrews. On the role of mobility for multimessage gossip. *IEEE Transactions on Information Theory*, 59(6):3953–3970, 2013.
8. Giuseppe Antonio Di Luna and Roberto Baldoni. Investigating the cost of anonymity on dynamic networks. *CoRR*, abs/1505.03509, 2015. URL: `http://arxiv.org/abs/1505.03509`, `arXiv:1505.03509`.
9. Giuseppe Antonio Di Luna and Roberto Baldoni. Non trivial computations in anonymous dynamic networks. In *Proceedings of the 19th International Conference on Principles of Distributed Systems*, Leibniz International Proceedings in Informatics, 2015. To appear.
10. Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Conscious and unconscious counting on anonymous dynamic networks. In *Proceedings of the 15th International Conference on Distributed Computing and Networking*, volume 8314 of *Lecture Notes in Computer Science*, pages 257–271. Springer Berlin Heidelberg, 2014.
11. Giuseppe Antonio Di Luna, Roberto Baldoni, Silvia Bonomi, and Ioannis Chatzigiannakis. Counting in anonymous dynamic networks under worst-case adversary. In *Proceedings of the 34th International Conference on Distributed Computing Systems*, pages 338–347. IEEE, 2014.
12. Giuseppe Antonio Di Luna, Silvia Bonomi, Ioannis Chatzigiannakis, and Roberto Baldoni. Counting in anonymous dynamic networks: An experimental perspective. In *Proceedings of the 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, volume 8243 of *Lecture Notes in Computer Science*, pages 139–154. Springer Berlin Heidelberg, 2014.
13. M. Farach-Colton, A. Fernández Anta, A. Milani, M. A. Mosteiro, and S. Zaks. Opportunistic information dissemination in mobile ad-hoc networks: adaptiveness vs. obliviousness and randomization vs. determinism. In *Proc. of the 10th Latin American Theoretical Informatics Symposium*, volume 7256 of *Lecture Notes in Computer Science*, pages 303–314. Springer-Verlag, Berlin, 2012.

14 Martin Farach-Colton and Miguel A. Mosteiro. Initializing sensor networks of non-uniform density in the weak sensor model. *Algorithmica*, 73(1):87–114, 2015. doi:10.1007/s00453-014-9905-5.

15 A. Fernández Anta and M. A. Mosteiro. Contention resolution in multiple-access channels: k-selection in radio networks. *Discrete Mathematics, Algorithms and Applications*, 02(04):445–456, 2010. doi:10.1142/S1793830910000796.

16 Antonio Fernández Anta, Alessia Milani, Miguel A. Mosteiro, and Shmuel Zaks. Opportunistic information dissemination in mobile ad-hoc networks: the profit of global synchrony. *Distributed Computing*, 25(4):279–296, 2012. doi:10.1007/s00446-012-0165-9.

17 D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of the 44th IEEE Ann. Symp. on Foundations of Computer Science*, pages 482–491, 2003.

18 E. Kranakis, D. Krizanc, and J. Vandenberg. Computing boolean functions on anonymous networks. *Information and Computation*, 114(2):214–236, 1994. doi:10.1006/inco.1994.1086.

19 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 513–522. ACM, 2010.

20 Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Naming and counting in anonymous unknown dynamic networks. In *Stabilization, Safety, and Security of Distributed Systems*, pages 281–295. Springer, 2013.

21 Alessia Milani and Miguel A. Mosteiro. A faster counting protocol for anonymous dynamic networks. In *Proceedings of the 19th International Conference on Principles of Distributed Systems*, volume 46 of *Leibniz International Proceedings in Informatics*, pages 1–13, 2015.

22 Damon Mosk-Aoyama and Devavrat Shah. Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 54(7):2997–3007, 2008.

23 Angelia Nedic, Alex Olshevsky, Asuman Ozdaglar, and John N Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

24 Sujay Sanghavi, Bruce Hajek, and Laurent Massoulié. Gossiping with multiple messages. *IEEE Transactions on Information Theory*, 53(12):4640–4654, 2007.