Global Computation in a Poorly Connected World: Fast Rumor Spreading with No Dependence on Conductance

Keren Censor-Hillel MIT
Bernhard Haeupler MIT
Jonathan A. Kelner MIT
Petar Maymounkov MIT

Abstract. In this paper, we study the question of how efficiently a collection of interconnected nodes can perform a global computation in the widely studied $\mathcal{GOSSIP}$ model of communication. In this model, nodes do not know the global topology of the network, and they may only initiate contact with a single neighbor in each round. This model contrasts with the much less restrictive $\mathcal{LOCAL}$ model, where a node may simultaneously communicate with all of its neighbors in a single round. A basic question in this setting is how many rounds of communication are required for the information dissemination problem, in which each node has some piece of information and is required to collect all others.

In the $\mathcal{LOCAL}$ model, this is quite simple: each node broadcasts all of its information in each round, and the number of rounds required will be equal to the diameter of the underlying communication graph. In the $\mathcal{GOSSIP}$ model, each node must independently choose a single neighbor to contact, and the lack of global information makes it difficult to make any sort of principled choice. As such, researchers have focused on the *uniform gossip algorithm*, in which each node independently selects a neighbor uniformly at random. When the graph is well-connected, this works quite well. In a string of beautiful papers, researchers proved a sequence of successively stronger bounds on the number of rounds required in terms of the conductance ϕ, culminating in a bound of $O(\phi^{-1} \log n)$.

In this paper, we show that a fairly simple modification of the protocol gives an algorithm that solves the information dissemination problem in at most $O(D + \text{polylog}(n))$ rounds in a network of diameter D, with *no dependence on the conductance*. This is at most an additive polylogarithmic factor from the trivial lower bound of D, which applies even in the $\mathcal{LOCAL}$ model.

In fact, we prove that something stronger is true: *any* algorithm that requires T rounds in the $\mathcal{LOCAL}$ model can be simulated in $O(T + \text{polylog}(n))$ rounds in the $\mathcal{GOSSIP}$ model. We thus prove that these two models of distributed computation are essentially equivalent.

This work was partially supported by the Simons Postdoctoral Fellows Program and NSF grant CCF-0843915.

1. Introduction

Many distributed applications require nodes of a network to perform a global task using only local knowledge. Typically a node initially only knows the identity of its neighbors and gets to know a wider local neighborhood in the underlying communication graph by repeatedly communicating with its neighbors. Among the most important questions in distributed computing is how certain global computation problems, e.g., computing a maximal independent set [**21**] or a graph coloring [**2**], can be performed with such local constraints.

Many upper and lower bounds for distributed tasks are given for the well-known $\mathcal{LOCAL}$ model [**24**, Chapter 2], which operates in synchronized rounds and allows each node in each round to exchange messages of unbounded size with all of its neighbors. It is fair to say that the $\mathcal{LOCAL}$ model is essentially the established minimal requirement for a distributed algorithm. Indeed, whenever a distributed algorithm is said to have running time T it is implied that, at the least, there exists a T-round algorithm in the $\mathcal{LOCAL}$ model.

In many settings, practical system design or physical constraints do not allow a node to contact all of its (potentially very large number of) neighbors at once. In this paper we focus on this case and consider the $\mathcal{GOSSIP}$ model, which restricts each node to initiate at most one (bidirectional) communication with one of its neighbors per round. In contrast to computations in the $\mathcal{LOCAL}$ model, algorithms for the $\mathcal{GOSSIP}$ model have to decide which neighbor to contact in each round. This is particularly challenging when the network topology is unknown.

Algorithms with such *gossip constraints* have been intensively studied for the so-called Rumor problem (also known as the *rumor spreading* or *information dissemination* problem), in which each node has some initial input and is required to collect the information of all other nodes. Most previous papers analyzed the simple `UniformGossip` algorithm, which chooses a random neighbor to contact in each round. The uniform gossip mixes well on well-connected graphs, and good bounds for its convergence in terms of the graph conductance have been given [**4**, **13**, **23**]. However, it has a tendency to repeatedly communicate between well-connected neighbors while not transmitting information across bottlenecks. Only recently have algorithms been designed that try to avoid this behavior. By alternating between random and deterministic choices, [**3**] showed that fast convergence can be achieved for a wider family of graphs, namely, those which have large *weak conductance* (a notion defined therein). However, while this outperformed existing techniques in many cases, its running time bound still depended on a notion of the connectivity of the graph.

1.1. Our results. This paper significantly improves upon previous algorithms by providing the first information spreading algorithm for the $\mathcal{GOSSIP}$ model that is fast for *all* graphs, with no dependence on their conductance. Our algorithm requires at most $O(D + \text{polylog}(n))$ rounds in a network of size n and diameter D. This is at most an additive polylogarithmic factor from the trivial lower bound of $O(D)$ rounds even for the $\mathcal{LOCAL}$ model. In contrast, there are many graphs with polylogarithmic diameter on which all prior algorithms have $\Omega(n)$ bounds.

In addition, our results apply more generally to any algorithm in the $\mathcal{LOCAL}$ model. We show how any algorithm that takes T time in the $\mathcal{LOCAL}$ model can be simulated in the $\mathcal{GOSSIP}$ model in $O(T + \text{polylog}(n))$ time, thus incurring only an additional polylogarithmic cost in the size of the network n. Our main result that leads to this simulation is an algorithm for the $\mathcal{GOSSIP}$ model in which each node exchanges information (perhaps indirectly) with each of its neighbors within a polylogarithmic number of rounds. This holds for every graph, despite the possibility of large degrees. A key ingredient in this algorithm is a recursive decomposition of graphs into clusters of sufficiently large conductance, allowing fast (possibly indirect) exchange of information between nodes inside clusters. The decomposition guarantees that the number of edges between pairs of nodes that did not exchange information decreases by a constant fraction. To convert the multiplicative polylogarithmic overhead for each simulated round into the additive overhead in our final simulation result we show connections between sparse graph spanners and algorithms in the $\mathcal{GOSSIP}$ model. This allows us to simulate known constructions of nearly-additive sparse spanners [**26**], which then in turn can be used in our simulations for even more efficient communication.

1.2. Our Techniques. The key step in our approach is to devise a distributed subroutine in the $\mathcal{GOSSIP}$ model to efficiently simulate one round of the $\mathcal{LOCAL}$ model by a small number of $\mathcal{GOSSIP}$ rounds. In particular, the goal is to deliver each node's current messages to all of its neighbors, which we refer to as the NeighborExchange problem. Indeed, we exhibit such an algorithm, called `Superstep`, which requires at most $O(\log^3 n)$ rounds in the $\mathcal{GOSSIP}$ model for *all* graphs:

Theorem 1.1. *The* `Superstep` *algorithm solves* NeighborExchange *in the $\mathcal{GOSSIP}$ model in $O(\log^3 n)$ rounds.*

Our design for the `Superstep` algorithm was inspired by ideas from [**3**] and started with an attempt to analyze the following very natural algorithm for the NeighborExchange problem: In each round each node contacts a random neighbor whose message is not yet known to it. While this algorithm works well on most graphs, there exist graphs on which it requires a long time to complete due to asymmetric propagation of messages. We give an explicit example and discuss this issue in Section 6.

The `Superstep` algorithm is simple and operates by repeatedly performing $\log^3 n$ rounds of the `UniformGossip` algorithm, where each node chooses a random neighbor to contact at each round, followed by a reversal of the message exchanges to maintain symmetry. From [**4**] or its strengthening [**14**], it is known that all pairs of vertices (and in particular all pairs of neighbors) that lie inside a high-conductance subset of the underlying graph exchange each other's messages within a single iteration. An existential graph decomposition result, given in Corollary 3.4, shows that for any graph there is a decomposition into high-conductance clusters with at least a constant fraction of intra-cluster edges. This implies that the number of remaining message exchanges required decreases by a constant factor in each iteration, which results in a logarithmic number of iterations until NeighborExchange is solved.

This gives a simple algorithm for solving the Rumor problem, which requires all nodes to receive the messages of all other nodes: By iterating `Superstep` D times, where D is the diameter of the network, one obtains an $O(D \cdot \log^3 n)$ round algorithm. This is at most an $O(\log^3 n)$-factor slower than the trivial diameter lower bound and is a drastic improvement compared to prior upper bounds [**3**,**4**,**13**,**23**], which can be of order $O(n)$ even for networks with constant or logarithmic D.

Beyond the Rumor problem, it is immediate that the NeighborExchange problem bridges the gap between the $\mathcal{LOCAL}$ and $\mathcal{GOSSIP}$ models in general. Indeed, we can simply translate a single round of a $\mathcal{LOCAL}$ algorithm into the $\mathcal{GOSSIP}$ model by first using any algorithm for NeighborExchange to achieve the local broadcast and then performing the same local computations. We call this a simulation and more generally define an $(\alpha(G), \beta(G))$-*simulator* as a transformation that takes any algorithm in the $\mathcal{LOCAL}$ model that runs in $T(G)$ rounds if the underlying topology is G, and outputs an equivalent algorithm in the $\mathcal{GOSSIP}$ model that runs in $O_n(\alpha(G)) \cdot T(G) + O_n(\beta)$ rounds. Thus, the simulation based on the `Superstep` algorithm gives a $(\log^3 n, 0)$-simulator.

In many natural graph classes, like graphs with bounded genus or excluded minors, one can do better. Indeed we give a simple argument that on any (sparse) graph with *hereditary density* δ there is a schedule of direct message exchanges such that NeighborExchange is achieved in 2δ rounds. Furthermore an order-optimal schedule can be computed in $\delta \log n$ rounds of the $\mathcal{GOSSIP}$ model even if δ is not known. This leads to a $(\delta, \delta \log n)$-simulator.

Another way to look at this is that communicating over any hereditary sparse graph remains fast in the $\mathcal{GOSSIP}$ model. Thus, for a general graph, if one knows a sparse subgraph that has short paths from any node to its neighbors, one can solve the NeighborExchange problem by communicating via these paths. Such graphs have been intensely studied and are known as spanners. We show interesting connections between simulators and spanners. For one, any fast algorithm for the NeighborExchange problem induces a sparse low-stretch spanner. The `Superstep` algorithm can thus be seen as a new spanner construction in the $\mathcal{GOSSIP}$ model with the interesting property that the total number of messages used is at most $O(n \log^3 n)$. To our knowledge this is the first such construction. This also implies that, in general, NeighborExchange requires a logarithmic number of rounds (up to $\log\log n$ factors perhaps) in the $\mathcal{GOSSIP}$ model. Considering in the other direction, we show that any fast spanner construction in the $\mathcal{LOCAL}$ model can be used to further decrease the multiplicative overhead of our $(\log^3 n, 0)$-simulator. Applying this insight to several known spanner constructions [**6**, **10**, **26**, **27**] leads to our second main theorem:

Theorem 1.2. *Every algorithm in the $\mathcal{LOCAL}$ model which completes in $T = T(G)$ rounds when run on the topology G can be simulated in the $\mathcal{GOSSIP}$ model in*

$$O(1)\cdot\min\left\{T\cdot\log^3 n, T\cdot 2^{\log^* n}\log n + \log^4 n, T\cdot\log n + 2^{\log^* n}\log^4 n, T + \log^{O(1)} n, T\cdot\delta + \delta\log n, T\cdot\Delta\right\}$$

rounds, where n is the number of nodes, Δ the maximum degree and δ the hereditary density of G.

When we apply this result to the greedy algorithm for the Rumor problem, where $T = D$, we obtain an algorithm whose $O(D + \text{polylog}n)$ rounds are optimal up to the additive polylogarithmic term, essentially closing the gap to the known trivial lower bound of $\Omega(D)$.

1.3. Related Work. The problem of spreading information in a distributed system was introduced by Demers et al. [**5**] for the purpose of replicated database maintenance, and it has been extensively studied thereafter.

One fundamental property of the distributed system that affects the number of rounds required for information spreading is the communication model. The *random phone call* model was introduced by Karp et al. [**15**], allowing every node to contact one other node in each round. In our setting, this corresponds to the complete graph. This model alone received much attention, such as in bounding the number of calls [**7**], bounding the number of random bits used [**14**], bounding the number of bits [**12**], and more.

The number of rounds it takes to spread information for the randomized algorithm `UniformGossip`, in which every node chooses its communication partner for the next round uniformly at random from its set of neighbors, was analyzed using the conductance of the underlying graph by Mosk-Aoyama and Shah [**23**], by Chierichetti et al. [**4**], and later by Giakkoupis [**13**], whose work currently has the best bound in terms of conductance, of $O(\frac{\log n}{\Phi(G)})$ rounds, with high probability.

Apart from the uniform randomized algorithm, additional algorithms were suggested for spreading information. We shortly overview some of these approaches. Doerr et al. [**8**] introduce *quasi-random* rumor spreading, in which a node chooses its next communication partner by deterministically going over its list of neighbors, but the starting point of the list is chosen at random. Results are $O(\log n)$ rounds for a complete graph and the hypercube, as well as improved complexities for other families of graphs compared to the randomized rumor spreading algorithm with uniform distribution over neighbors. This was followed by further analysis of the quasi-random algorithm [**9**, **11**]. A hybrid algorithm, alternating between deterministic and randomized choices [**3**], was shown to achieve information spreading in $O(c(\frac{\log n}{\Phi_c(G)} + c))$ round, w.h.p., where $\Phi_c(G)$ is the *weak conductance* of the graph, a measure of connectivity of subsets in the graph. Distance-based bounds were given for nodes placed with uniform density in R^d [**16**,**17**], which also address gossip-based solutions to specific problems such as resource location and minimum spanning tree.

The $\mathcal{LOCAL}$ model of communication, where each node communicates with each of its neighbors in every round, was formalized by Peleg [**24**]. Information spreading in this model requires a number of rounds which is equal to the diameter of the communication graph. Many other distributed tasks have been studied in this model, and below we mention a few in order to give a sense of the variety of problems studied. These include computing maximal independent sets [**1**], graph colorings [**2**], computing capacitated dominating sets [**19**], general covering and packing problems [**20**], and general techniques for distributed symmetry breaking [**28**].

2. Preliminaries and Definitions

2.1. The `UniformGossip` Algorithm. The `UniformGossip` algorithm is a common algorithm for Rumor. (It is also known as the PUSH-PULL algorithm in some papers, such as [**13**].) Initially each vertex u has some message M_u. At each step, every vertex chooses a random incident edge (u, v) at which point u and v exchange all messages currently known to them. The process stops when all vertices know everyone's initial messages. In order to treat this process formally, for any fixed vertex v and its message M_v, we treat the set of vertices that know M_v as a set that evolves probabilistically over time, as we explain next.

We begin by fixing an ambient graph $G = (E, V)$, which is unweighted and directed. The `UniformGossip` process is a Markov chain over 2^V, the set of vertex subsets of G. Given a current state $S \subseteq V$, one transition is defined as follows. Every vertex u picks an incident outgoing edge $a_u = (u, w) \in E$ uniformly at random from all such candidates. Let us call the set of all chosen edges $A = \{a_u : u \in V\}$ an *activated set*. Further let $A^\circ = \{(u, w) : (u, w) \in A \text{ or } (w, u) \in A\}$ be the symmetric closure of A. The new state of the chain is given by $S \cup B$, where by definition a vertex v is in the *boundary* set B if and only if there exists $u \in S$ such that $(u, v) \in A^\circ$. Note that V is the unique absorbing state, assuming a non-empty start.

We say that an edge (u, w) is *activated* if $(u, w) \in A^\circ$. If we let S model the set of nodes in possession of the message M_v of some fixed vertex v and we assume bidirectional message exchange along activated edges, the new state $S \cup B$ (of the Markov process) actually describes the set of nodes in possession of the message M_v after one distributed step of the `UniformGossip` algorithm.

Consider a τ-step Markov process K, whose activated sets at each step are respectively $A_1, \dots, A_\tau$. Let the *reverse* of K, written K^{rev}, be the τ-step process defined by the activated sets $A_\tau, \dots, A_1$, in this order. For a process K, let $K(S)$ denote the end state when started from S.

Without loss of generality, for our analysis we will assume that only a single "starting" vertex s has an initial message M_s. We will be interested in analyzing the number of rounds of `UniformGossip` that ensure that all other vertices learn M_s, which we call the *broadcast time*. Clearly, when more than one vertex has an initial message, the broadcast time is the same since all messages are exchanged in parallel.

Lemma 2.1 (Reversal Lemma). *If $u \in K(\{w\})$, then $w \in K^{\text{rev}}(\{u\})$.*

In communication terms, the lemma says that if u receives a message originating at w after τ rounds determined by K, then w will receive a message originating at u after τ rounds determined by K^{rev}.

Proof. The condition $u \in K(\{w\})$ holds if and only if there exists a sequence of edges $(e_{i_1}, \dots, e_{i_r})$ such that $e_{i_j} \in A^\circ_{i_j}$ for all j, the indices are increasing in that $i_1 < \cdots < i_r$, and the sequence forms a path from w to u. The presence of the reversed sequence in K^{rev} implies $w \in K^{\text{rev}}(\{u\})$. □

2.2. Conductance. The notion of *graph conductance* was introduced by Sinclair [**29**]. We require a more general version, which we introduce here. We begin with the requisite notation on edge-weighted graphs. We assume that each edge (u, v) has a weight $w_{uv} \in [0, 1]$. For an unweighted graph $G = (V, E)$ and any $u, v \in V$, we define $w_{uv} = 1$ if $(u, v) \in E$ and $w_{uv} = 0$ otherwise. Now we set $w(S, T) = \sum_{u \in S, v \in T} w_{uv}$. Note that in this definition it need not be the case that $S \cap T = \emptyset$, so, e.g., $w(S, S)$, when applied to an unweighted graph, counts every edge in S twice. The volume of a set $S \subseteq V$ with respect to V is written as $\mathrm{vol}(S) = w(S, V)$. Sometimes we will have different graphs defined over the same vertex set. In such cases, we will write the identity of the graph as a subscript, as in $\mathrm{vol}_G(S)$, in order to clarify which is the ambient graph (and hence the ambient edge set). Further, we allow self-loops at the vertices. A single loop at v of weight α is modeled by setting $w_{vv} = 2\alpha$, because both ends of the edge contribute α.

For a graph $G = (V, E)$ and a cut (S, T) where $S, T \subseteq V$ and $S \cap T = \emptyset$ (but where $T \cup S$ does not necessarily equal all of V), the *cut conductance* is given by

$$\varphi(S, T) = \frac{w(S, T)}{\min \{ \mathrm{vol}_G(S), \mathrm{vol}_G(T) \}}. \tag{1}$$

For a subset $H \subseteq V$ we need to define the *conductance of H (embedded) in V*. We will use this quantity to measure how quickly the `UniformGossip` algorithm proceeds in H, while accounting for the fact that edges in $(H, V - H)$ may slow down the process. The conductance of H in G is defined by

$$\Phi(H) = \min_{S \subseteq H} \varphi(S, H - S) \tag{2}$$

Note that the classical notion of conductance of G (according to Sinclair [**29**]) equals $\Phi(V)$ in our notation.

When we want to explicitly emphasize the ambient graph G within which H resides, we will write $\Phi_G(H)$.

A few arguments in this paper will benefit from the notion of a "strongly induced" graph of a vertex subset of an ambient graph G.

Definition 2.2. *Let $U \subseteq V$ be a vertex subset of G. The strongly induced graph of U in G is a (new) graph H with vertex set U, whose edge weight function $h : U \times U \to \mathbf{R}$ is defined by*

$$h_{uv} = \begin{cases} w_{uv}, & \text{if } u \neq v, \\ w_{uu} + \sum_{x \in V - U} w_{ux}, & \text{if } u = v. \end{cases}$$

Note that by construction we have $\Phi_H(U) = \Phi_G(U)$. The significance of this notion is the fact that the Markov process, describing the vertex set in possession of some message M_s for a starting vertex $s \in U$ in the `UniformGossip` algorithm executed on the strongly induced H, behaves identically to the respective process in G observed only on U. In particular, this definition allows us to use Theorem 1 of [**13**] in the following form:

Theorem 2.3. *For any graph $G = (V, E)$ and a subgraph $U \subseteq V$ and any start vertex in U, the broadcast time of the* `UniformGossip` *algorithm on U is $O(\Phi_G(U)^{-1} \log |U|)$ rounds w.h.p.*

3. Solving NeighborExchange in $O(\log^3 n)$ Rounds

The idea behind our algorithm for solving the NeighborExchange problem is as follows. For every graph there exists a partition into clusters whose conductance is high, and therefore the `UniformGossip` algorithm allows information to spread quickly in each cluster. The latter further implies that pairs of neighbors inside a cluster exchange their messages quickly (perhaps indirectly). What remains is to exchange messages across inter-cluster edges. This is done recursively. In the following subsection we describe the conductance decomposition and then in Subsection 3.2 we give the details for the algorithm together with the proof of correctness.

3.1. Conductance Decomposition of a Graph. As described, our first goal is to partition the graph into clusters with large conductance. The challenge here is to do so while limiting the number of inter-cluster edges, so that we can efficiently apply this argument recursively. (Otherwise, this could be trivially done in any graph, for example by having each node as a separate cluster.) We are going to achieve this in the following lemma whose proof (found in Appendix A) is very similar to that of Theorem 7.1 in [**30**]. Note that for our eventual algorithm, we are only going to need an existential proof of this clustering and not an actual algorithm for finding it.

Lemma 3.1. *Let $S \subseteq V$ be of maximum volume such that $\operatorname{vol}(S) \le \operatorname{vol}(V)/2$ and $\varphi(S, V-S) \le \xi$, for a fixed parameter $\xi \ge \Phi(G)$. If $\operatorname{vol}(S) \le \operatorname{vol}(V)/4$, then $\Phi(V-S) \ge \xi/3$.*

Lemma 3.1 says that if a graph has no sparse balanced cuts, then it has a large subgraph which has no sparse cuts. The following corollary establishes that Lemma 3.1 holds even in the case when the ambient graph is itself a subgraph of a larger graph.

Corollary 3.2. *Let $U \subseteq V$ and let $S \subseteq U$ be of maximum volume such that $\operatorname{vol}(S) \le \operatorname{vol}(U)/2$ and $\varphi(S, U-S) \le \xi$, for a fixed parameter $\xi \ge \Phi(U)$. If $\operatorname{vol}(S) \le \operatorname{vol}(U)/4$, then $\Phi(U-S) \ge \xi/3$.*

Proof. Observe that the proof of Lemma 3.1 holds when the graph has loops, i.e. $w_{uu} \neq 0$ for some u's. Let H be the strongly induced graph of U. It follows from the definition that for any two disjoint sets $A, B \subseteq U$ we have $\operatorname{vol}_G(A) = \operatorname{vol}_H(A)$ and $w(A,B) = h(A,B)$. We can therefore apply Lemma 3.1 to H and deduce that the statement holds for the respective sets in G. □

We are now ready to state and analyze the strong clustering algorithm. We emphasize that this is not a distributed algorithm, but an algorithm that only serves as a proof of existence of the partition. First, consider the following subroutine:

```
Cluster($G$,$U$,$\xi$):
    The inputs are a graph $G = (V, E)$, a subset $U \subseteq V$ and a parameter $0 < \xi < 1$.
  1. Find a subset $S \subseteq U$ of maximum volume such that $\operatorname{vol}(S) \le \operatorname{vol}(U)/2$ and $\varphi(S, U-S) \le \xi$.
  2. If no such $S$ exists, then stop and output a single cluster $\{U\}$. Otherwise,
  3a. If $\operatorname{vol}(S) \le \operatorname{vol}(U)/4$, output $\{U-S\} \cup$ Cluster($G$,$S$,$\xi$).
  3b. If $\operatorname{vol}(S) > \operatorname{vol}(U)/4$, output Cluster($G$,$S$,$\xi$) $\cup$ Cluster($G$,$U-S$,$\xi$).
```

The clustering algorithm for a graph $G = (V, E)$ is simply a call to `Cluster(`G`,`V`,`ξ`)`. The following theorem is proven in Appendix B.

Theorem 3.3. *For every $0 < \zeta < 1$, every graph $G = (V, E)$ with edge weights $w_{uv} \in \{0\} \cup [1, +\infty)$ has a partition $V = V_1 \cup \cdots \cup V_k$ such that $\Phi(V_i) \ge \frac{\zeta}{\log_{4/3} \operatorname{vol}(V)}$, for all i, and $\sum_{i<j} w(V_i, V_j) \le \frac{3\zeta}{2} \operatorname{vol}(V)$.*

In this paper, we are going to use the following specialization of this theorem, obtained by plugging in $\zeta = 1/3$:

Corollary 3.4. *Every unweighted graph on m edges has a clustering that cuts at most $\frac{m}{2}$ edges and each cluster has conductance at least $\frac{1}{3 \log_{4/3} 2m}$.*

3.2. The `Superstep` Algorithm for the NeighborExchange Problem. In this section, we will the describe the `Superstep` algorithm, which solves the NeighborExchange problem. Recall that, for this problem, all vertices v are assumed to possess an initial message M_v, and the goal is for every pair of neighbors to know each other's initial messages.

We now describe our communication protocol, which specifies a local, per-vertex rule that tells a node which edge to choose for communication at any given round. It is assumed that the node will greedily transmit all messages known to it whenever an edge is chosen for communication. The protocol described here will employ some auxiliary messages, which are needed exclusively for its internal workings.

The `Superstep` subroutine described in this section is designed to ensure that, after a single invocation, all neighbors (u, w) in an undirected graph G have exchanged each other's initial messages. Clearly then, D invocations of `Superstep`, where D is the diameter of G, ensure that a message starting at vertex v reaches all $u \in V$, and this holds for all messages. D invocations of `Superstep` thus resolve the Rumor problem.

If E is a set of undirected edges, let $\vec{E} = \{(u, w) : \{u, w\} \in E\}$ be the corresponding directed graph.

```
Superstep(G,τ):
The parameter $G = (V, E)$ is an unweighted, undirected graph, and $\tau$ is a positive integer.
Set $F_0 := \vec{E}$ and $i := 0$. While $F_i \neq \emptyset$, repeat:
  1. (First half)
      1a. Initialize every vertex $v$ with a new auxiliary message $a(v)$, unique to $v$.
          (This messages is added to the set of initial messages that $v$ happens to know
          currently.)
      1b. Perform the UniformGossip algorithm with respect to $F_i$ for $\tau$ rounds. And
          denote the outcome of the random activated edge choices by $K_i$
      1c. For every vertex $u$ and neighbor $w$, let $X_{uw}$ be the indicator that $u$ received
          $a(w)$
  2. (Second half)
      2a. Initialize every vertex $v$ with a fresh auxiliary message $b(v)$, unique to $v$
      2b. Perform $K_i^{\mathrm{rev}}$, the reverse process of the one realized in Step 1b
      2c. For every vertex $u$ and neighbor $w$, let $Y_{uw}$ be the indicator that $u$ received
          $b(w)$
  3. (Pruning) Compute the set of pruned directed edges $P_i = \{(u, w) : X_{uw} + Y_{uw} > 0\}$
  4. Set $F_{i+1} := F_i - P_i$ and $i := i + 1$
```

It is easily verified that the above algorithm can be implemented in the $\mathcal{GOSSIP}$ model of communication.

Theorem 3.5. *Let $G = (V, E)$ be an undirected, unweighted graph with $|V| = n$ and $|E| = m$. Then, after one invocation of* `Superstep(G,τ)`*, where $\tau = \Theta(\log^2 m)$, the following hold with probability $1 - 1/n^{\Omega(1)}$:*

(i) *Every pair of neighbors $\{u, w\} \in E$ receive each other's messages.*
(ii) *The algorithm performs $\Theta(\log^3 m)$ distributed rounds.*

Our proof of Theorem 3.5 is structured as follows. Let $\vec{E} = F_0, \ldots, F_d = \emptyset$ be the respective edge sets of each iteration in `Superstep`. We are going to show that, with probability $1 - 1/n^{\Omega(1)}$, the following invariants are maintained at each iteration:

(a) The directed edge set F_i is symmetric in the sense that $(u, w) \in F_i \Rightarrow (w, u) \in F_i$,
(b) The size of F_i reduces by a constant factor at each iteration. Formally, $\mathrm{vol}(F_{i+1}) \leq \frac{1}{2}\,\mathrm{vol}(F_i)$, and
(c) After the i-th iteration, for every $(u, w) \in \vec{E} - F_{i+1}$, vertex u has received the message of vertex w and vice-versa.

Since $F_d = \emptyset$, claim (c) implies part (i) of Theorem 3.5. Claim (b) implies that the maximum number of iterations is $\log 2m$. Noting that every iteration entails 2τ distributed rounds, establishes part (ii) of Theorem 3.5.

PROOF OF CLAIM (a): Initially, F_0 is symmetric by construction. Inductively, assume that F_i is symmetric. The Reversal Lemma applied to K_i and K_i^{rev} implies $X_{uw} = Y_{wu}$, for all $u, w \in V$. This in turn implies that $X_{uw} + Y_{uw} = X_{wu} + Y_{wu}$, so P_i is symmetric. Since F_i is symmetric by hypothesis, we can conclude that $F_{i+1} = F_i - P_i$ is symmetric as well. □

PROOF OF CLAIM (b): Consider the graph $G_i = (V, F_i)$ on the edge set F_i. Since F_i is symmetric, by Claim (a), we can treat G_i as undirected for the purposes of analyzing the `UniformGossip` algorithm. Let $V_1 \cup \cdots \cup V_k$ be the decomposition of G_i promised by Corollary 3.4. (Note that the corollary holds for disconnected graphs, which may arise.) We thus have $\Phi(V_j) \geq \frac{1}{3 \log 4/32m}$, for all $1 \leq j \leq k$.

The choice $\tau = O\big(3 \log_{4/3} 2m \cdot \log m\big)$ ensures, via Theorem 2.3, that the first `UniformGossip` execution in every iteration mixes on all V_j with probability $1 - 1/n^{\Omega(1)}$. Mixing in V_j implies that for every internal edge (u, w), where $u, w \in V_j$ and $(u, w) \in F_i$, the vertices (u, w) receive each other's auxiliary messages. The latter is summarized as $X_{uw} = X_{wu} = 1$. Applying the Reversal Lemma to the second execution of the `UniformGossip` algorithm, we deduce that $Y_{uw} = Y_{wu} = 1$ as well. These two equalities imply, by the definition of P_i, that P_i is a superset of the edges not cut by the decomposition $V_1 \cup \cdots \cup V_k$. Equivalently, F_{i+1} is a subset of the cut edges. Corollary 3.4, however, bounds the volume of the cut edges by $\frac{1}{2} \operatorname{vol}(F_i)$, which concludes the proof of Claim (b). □

PROOF OF CLAIM (c): Initially, $\vec{E} - F_0 = \emptyset$ and so the claim holds trivially. By induction, the claim holds for edges in $\vec{E} - F_i$. And so it suffices to establish that u and v exchange their respective payload messages for all $(u, w) \in P_i$. However, this is equivalent to the conditions $X_{uw} + Y_{uw} > 0$, which are enforced by the definition of P_i. □

Finally, our main result, Theorem 1.1, follows as a corollary of Theorem 3.5.

4. Solving NEIGHBOREXCHANGE in Hereditary Sparse Graphs

Now we ask what can be achieved if instead of exchanging information indirectly as done in the `Superstep` algorithm, we exchange information only directly between neighbors. We will show in this section that this results in very simple deterministic algorithms for an important class of graphs that includes bounded genus graphs and all graphs that can be characterized by excluded minors [**18**, **22**]. The results here will be used for the more general simulators in Section 5.

As before we will focus on solving the NEIGHBOREXCHANGE problem. One trivial way to solve this problem is for each node to contact its neighbors directly, e.g., by using a simple round robin method. This takes at most Δ time, where Δ is the maximum-degree of the network. However, in some cases direct message exchanges work better. One graph that exemplifies this is the star graph on n nodes. While it takes $\Delta = n$ time to complete a round robin in the center, after just a single round of message exchanges each leaf has initiated a bidirectional link to the center and thus exchanged its messages. On the other hand, scheduling edges cannot be fast on dense graphs with many more edges than nodes. The following lemma shows that the *hereditary density* captures how efficient direct message exchanges can be on a given graph:

Lemma 4.1. *Let the hereditary density δ of a graph G be the minimal integer such that for every subset of nodes S the subgraph induced by S has at most density δ, i.e., at most $\delta|S|$ edges.*

(1) *Any schedule of direct message exchanges that solves the* NEIGHBOREXCHANGE *problem on G takes at least δ rounds.*
(2) *There exists a schedule of the edges of G such each node needs only 2δ direct message exchanges to solve the* NEIGHBOREXCHANGE *problem.*

PROOF. Since the hereditary density of G is δ, there is a subset of nodes $S \subseteq V$ with at least $\delta|S|$ edges between nodes in S. In each round, each of the $|S|$ nodes is allowed to schedule at most one message exchange, so a simple pigeonhole principle argument shows that at least one node needs to initiate at least δ message exchanges.

For the second claim, we are going to show that for any $\epsilon > 0$ there is an $O(\epsilon^{-1} \log n)$-time deterministic distributed algorithm in the $\mathcal{LOCAL}$ model that assigns the edges of G to nodes such that each node is assigned at most $2(1+\epsilon)\delta$ edges. Then setting $\epsilon < (3\delta)^{-1}$ makes the algorithm inefficient but finishes the existential proof.

The algorithm runs in phases in which, iteratively, a node takes responsibility for some of the remaining edges connected to it. All edges that are assigned are then eliminated and so are nodes that have no unassigned incident edges. In each phase, every node of degree at most $2(1+\epsilon)\delta$ takes responsibility for all of its incident edges (breaking ties arbitrarily). At least a $1/(1+\frac{1}{\epsilon})$ fraction of the remaining nodes fall under this category in every phase. This is because otherwise, the number of edges in the subgraph would be more than $(|S| - |S|/(1+\frac{1}{\epsilon}))(2(1+\epsilon)\delta)/2 = |S|\delta$, which would contradict the fact thatthe hereditary densityof the graph equals δ of. What remains after each phase is an induced subgraph which, by definition of the hereditary density, continues to have hereditary density at most δ. The number of remaining nodes thus decreases by a factor of $1 - 1/(1+\frac{1}{\epsilon})$ in every phase and it takes at most $O(\log_{1+\epsilon} n)$ phases until no more nodes remain, at which point all edges have been assigned to a node. $\square$

We note that the lower bound of Lemma 4.1 is tight in all graphs, i.e., the upper bound of 2δ can be improved to δ. Graphs with hereditary density δ, also known as $(0,\delta)$-sparse graphs, are thus exactly the graphs in which δ is the minimum number such that the edges can be oriented to form a directed graph with outdegree at most δ. This in turn is equivalent to the *pseudoarboricity* of the graph, i.e., the minimum number of pseudoforests needed to cover the graph. Due to the matroid structure of pseudoforests, the pseudoarboricity can be computed in polynomial time. For our purposes the (non-distributed) algorithms to compute these optimal direct message exchange schedule are too slow. Instead, we present a simple and fast algorithm, based on the $\mathcal{LOCAL}$ algorithm in Lemma 4.1, which computes a schedule that is within a factor of $2+\epsilon$ of the optimal. We note that the `DirectExchange` algorithm presented here works in the $\mathcal{GOSSIP}$ model and furthermore does not require the hereditary density δ to be known *a priori*. The following is the algorithm for an individual node v:

```
DirectExchange:
Set $\delta' = 1$ and $H = \emptyset$. $H$ is the subset of neighbors in $\Gamma(v)$ that node $v$ has exchanged
messages with. Repeat:
    $\delta' = (1+\epsilon)\delta'$
    for $O(\frac{1}{\epsilon} \cdot \log n)$ rounds do
        if $|\Gamma(v) \setminus H| \leq \delta'$
            during the next $\delta'$ rounds exchange messages with all neighbors in $\Gamma(v) \setminus$
            $H$
            terminate
        else
            wait for $\delta'$ rounds
        update $H$
```

Theorem 4.2. *For any constant $\epsilon > 0$, the deterministic algorithm* `DirectExchange` *solves the* NEIGHBOREXCHANGE *problem in the $\mathcal{GOSSIP}$ model using $O(\frac{\delta \log n}{\epsilon^2})$ rounds, where δ is the hereditary density of the underlying topology. During the algorithm, each node initiates at most $2(1+\epsilon)^2\delta$ exchanges.*

Proof. Let δ be the hereditary density of the underlying topology. We know from the proof of Lemma 4.1 that the algorithm terminates during the for-loop if δ' is at least $2(1+\epsilon)\delta$. Thus, when the algorithm terminates, δ' is at most $2(1+\epsilon)^2\delta$ which is also an upper bound on the number of neighbors contacted by any node. In the $(i+1)^{\text{th}}$-to-last iteration of the outer loop, δ' is at most $2(1+\epsilon)^2\delta/(1+\epsilon)^i$, and the running time for this phase is thus at most $2(1+\epsilon)^2\delta/(1+\epsilon)^i \cdot O(\frac{1}{\epsilon}\log n)$. Summing up over these powers of $1/(1+\epsilon)$ results in a total of at most $\delta/((1+\epsilon)-1)\cdot O(\frac{1}{\epsilon}\log n) = O(\frac{\delta \log n}{\epsilon^2})$ rounds. □

5. Simulators and Graph Spanners

In this section we generalize our results to arbitrary simulations of $\mathcal{LOCAL}$ algorithms in the $\mathcal{GOSSIP}$ model and point out connections to graph spanners, another well-studied subject.

Recall that we defined the NeighborExchange problem exactly in such a way that it simulates in the $\mathcal{GOSSIP}$ model what is done in one round of the $\mathcal{LOCAL}$ model. With our solutions, an $O(\delta \log n)$-round algorithm and an $O(\log^3 n)$-round algorithm for the NeighborExchange problem in the $\mathcal{GOSSIP}$ model, it is obvious that we can now easily convert any T-round algorithm for the $\mathcal{LOCAL}$ model to an algorithm in the $\mathcal{GOSSIP}$ model, e.g., by T times applying the `Superstep` algorithm. In the case of the `DirectExchange` algorithm we can do even better. While it takes $O(\delta \log n)$ rounds to compute a good scheduling, once it is known it can be reused and each node can simply exchange messages with the same $O(\delta)$ nodes without incurring an additional overhead. Thus, simulating the second and any further rounds can be easily done in $O(\delta)$ rounds in the $\mathcal{GOSSIP}$ model. This means that any algorithm that takes $O(T)$ rounds to complete in the $\mathcal{LOCAL}$ model can be converted to an algorithm that takes $O(\delta T + \delta \log n)$ rounds in the $\mathcal{GOSSIP}$ model. We call this a simulation and define simulators formally as follows.

Definition 5.1. *An (α, β)-simulator is a way to transform any algorithm A in the $\mathcal{LOCAL}$ model to an algorithm A' in the $\mathcal{GOSSIP}$ model such that A' computes the same output as A and if A takes $O(T)$ rounds than A' takes at most $O(\alpha T + \beta)$ rounds.*

Phrasing our results from Section 3.2 and Section 4 in terms of simulators we get the following corollary.

Corollary 5.2. *For a graph G of n nodes, hereditary density δ, and maximum degree Δ, the following hold:*

- *There is a randomized $(\log^3 n, 0)$-simulator.*
- *There is a deterministic $(\Delta, 0)$-simulator.*
- *There is a deterministic $(2(1+\epsilon)^2\delta, O(\delta\epsilon^{-2}\log n))$-simulator for any $\epsilon > 0$ or, simply, there is a $(\delta, \delta \log n)$-simulator.*

Note that for computations that require many rounds in the $\mathcal{LOCAL}$ model the $(2(1+\epsilon)^2\delta, O(\delta\epsilon^{-2}\log n))$-simulator is a $\log n$-factor faster than repeatedly applying the `DirectExchange` algorithm. This raises the question whether we can similarly improve our $(\log^3 n, 0)$-simulator to obtain a smaller multiplicative overhead for the simulation.

What we would need for this is to compute, e.g., using the `Superstep` algorithm, a schedule that can then be repeated to exchange messages between every node and its neighbors. What we are essentially asking for is a short sequence of neighbors for each node over which each node can indirectly get in contact with all its neighbors. Note that any such schedule of length t must at least fulfill the property that the union of all edges used by any node is connected (if the original graph G is connected) and even more that each node is connected to all its neighbors via a path of length at most t. Subgraphs with this property are called *spanners*. Spanners are well-studied objects, due to their extremely useful property that they approximately preserve distances while potentially being much sparser than the original graph. The quality of a spanner is described by two parameters, its number of edges and its *stretch*, which measures how well it preserves distances.

Definition 5.3 (Spanners)**.** *A subgraph $S = (V, E')$ of a graph $G = (V, E)$ is called an (α, β)-*stretch spanner *if any two nodes u, v with distance d in G have distance at most $\alpha d + \beta$ in S.*

From the discussion above it is also clear that any solution to the NEIGHBOREXCHANGE problem in the $\mathcal{GOSSIP}$ model also computes a spanner as a byproduct.

Lemma 5.4. *If A is an algorithm in the $\mathcal{GOSSIP}$ model that solves the* NEIGHBOREXCHANGE *problem in any graph G in T rounds then this algorithm can be used to compute a $(T, 0)$-stretch spanner with hereditary density T in $O(T)$ rounds in the $\mathcal{GOSSIP}$ model.*

While there are spanners with better properties than the $(\log^3 n, 0)$-stretch and $\log^3 n$-density implied by Lemma 5.4 and Theorem 3.5, our construction has the interesting property that the number of messages exchanged during the algorithm is at most $O(n \log^3 n)$, whereas all prior algorithms rely on the broadcast nature of the $\mathcal{LOCAL}$ model and therefore use already $O(n^2)$ messages in one round on a dense graph. Lemma 5.4 furthermore implies a nearly logarithmic lower bound on the time that is needed in the $\mathcal{GOSSIP}$ model to solve the NEIGHBOREXCHANGE problem:

Corollary 5.5. *For any algorithm in the $\mathcal{GOSSIP}$ model that solves the* NEIGHBOREXCHANGE *problem there is a graph G on n nodes on which this algorithm takes at least $\omega(\frac{\log n}{\log \log n})$ rounds,.*

PROOF. Assume an algorithm takes at most $T(n)$ rounds on any graph with n nodes. The edges used by the algorithm form a $T(n)$-stretch spanner with density $T(n)$, as stated in Lemma 5.4. For values of $T(n)$ which are too small it is known that such spanners do not exist [**25**]. More specifically it is known that there are graphs with n nodes, density at least $1/4n^{1/r}$ and girth r, i.e., the length of the smallest cycle is r. In such a graph any $(r-2)$-stretch spanner has to be the original graph itself, since removing a single edge causes its end-points to have distance at least $r-1$, and thus the spanner also have density $1/4n^{1/r}$. Therefore $T(n) \geq \text{argmin}_r \{r-2, 1/4n^{1/r}\} = \omega(\frac{\log n}{\log \log n})$. □

Interestingly, it is not only the case that efficient simulators imply good spanners but the next lemma shows as a converse that good existing spanner constructions for the $\mathcal{LOCAL}$ model can be used to improve the performance of simulators.

Theorem 5.6. *If there is an algorithm that computes an (α, β)-stretch spanner with hereditary density δ in $O(T)$ rounds in the $\mathcal{LOCAL}$ model than this can be combined with an (α', β')-simulator to an $(\alpha\delta, T\alpha' + \beta' + \delta \log n + \delta\beta)$-simulator.*

PROOF. For simplicity we first assume that $\beta = 0$, i.e., the spanner S computed by the algorithm in the $\mathcal{LOCAL}$ model has purely multiplicative stretch α and hereditary density δ. Our strategy is simple: We are first going to compute the good spanner by simulating the spanner creation algorithm from the $\mathcal{LOCAL}$ model using the given simulator. This takes $T\alpha' + \beta'$ rounds in the $\mathcal{GOSSIP}$ model. Once this spanner S is computed we are only going to communicate via the edges in this spanner. Note that for any node there is a path of length at most α to any of its neighbors. Thus if we perform α rounds of $\mathcal{LOCAL}$-flooding rounds in which each node forwards all messages it knows of to all its neighbors in S each node obtains the messages of all its neighbors in G. This corresponds exactly to a NEIGHBOREXCHANGE in G. Therefore if we want to simulate T' rounds of an algorithm A in the $\mathcal{LOCAL}$ model on G we can alternatively perform αT $\mathcal{LOCAL}$ computation rounds on S while doing the $\mathcal{LOCAL}$ computations of A every α rounds. This is a computation in the $\mathcal{LOCAL}$ model but on a sparse graph. We are therefore going to use the $(O(\delta), O(\delta \log n))$-simulator from Corollary 5.2 to simulate this computation which takes

$O(\delta\alpha T' + \delta \log n)$ rounds in the $\mathcal{GOSSIP}$ model. Putting this together with the $T\alpha' + \beta'$ rounds it takes to compute the spanner S we end up with $\delta\alpha T' + \delta \log n + T\alpha' + \beta'$ rounds in total.

In general (i.e., for $\beta > \alpha$) it is not possible (see, e.g., Corollary 5.5) to simulate the $\mathcal{LOCAL}$ algorithm step by step. Instead we rely on the fact that any $\mathcal{LOCAL}$ computation over T rounds can be performed by each node first gathering information of all nodes in a T-neighborhood and then doing $\mathcal{LOCAL}$ computations to determine the output. For this all nodes simply include all their initial knowledge (and for a randomized algorithm all the random bits they might use throughout the algorithm) in a message and flood this in T rounds to all node in their T-neighborhood. Because a node now knows all information that can influence its output over a T-round computation it can now locally simulate the algorithm for itself and its neighbors to the extend that its output can be determined. Having this we simulate the transformed algorithm as before: We first precomute S in $T\alpha' + \beta'$ time and then simulate the T' rounds of flooding in G by performing $\alpha T' + \beta$ rounds of $\mathcal{LOCAL}$-flooding in S. Using the $(O(\delta), O(\delta \log n))$-simulator this takes $O(\delta(\alpha T' + \beta) + \delta \log n)$ rounds in the $\mathcal{GOSSIP}$ model.

□

Corollary 5.7. *There is a* $(2^{\log^* n} \log n, \log^4 n)$*-simulator, a* $(\log n, 2^{\log^* n} \log^4 n)$*-simulator and a* $(O(1), \text{polylog} n)$*-simulator.*

Proof. We are going to construct the simulators with increasingly better multiplicative overhead by applying Theorem 5.6 to existing spanner constructions [**6**, **10**, **26**, **27**] for the $\mathcal{LOCAL}$ model. We first construct a $(\log^2 n, \log^4 n)$-simulator by combining our new $(\log^3 n, 0)$-simulator with the deterministic spanner construction in [**6**]. The construction in [**6**] takes $O(\log n)$ rounds in the $\mathcal{LOCAL}$ model and adds at most one edge to each node per round. Using $\alpha = T = \delta = O(\log n)$, $\alpha' = \log^3 n$ and $\beta = \beta' = 0$ in Theorem 5.6 gives the desired $(\log^2 n, \log^4 n)$-simulator. Having this simulator, we can use [**27**] to improve the multiplicative overhead while keeping the additive simulation overhead the same. In [**27**] an $\alpha = (2^{\log^* n} \log n)$-stretch spanner with constant hereditary density $\delta = O(1)$ is constructed in $T = O(2^{\log^* n} \log n)$-time in the $\mathcal{LOCAL}$ model. Using these parameters and the $(\log^2 n, \log^4 n)$-simulator in Theorem 5.6 leads to the strictly better $(2^{\log^* n} \log n, \log^4 n)$-simulator claimed here. Having this simulator, we can use it with the randomized spanner construction in [**10**]. There, an α-stretch spanner, with $\alpha = O(\log n)$, is constructed in $T = O(\log^3 n)$-time in the $\mathcal{LOCAL}$ model by extracting a subgraph with $\Omega(\log n)$ girth. Such a graph has constant hereditary density $\delta = O(1)$, as argued in [**25**]. Using these parameters and the $(2^{\log^* n} \log n, \log^4 n)$-simulator in Theorem 5.6 leads to the $(\log n, 2^{\log^* n} \log^4 n)$-simulator. Finally, we can use any of these simulators together with the nearly-additive $(5 + \epsilon, \text{polylog} n)$-spanner construction from [**26**] to obtain our last simulator. It is easy to verify that the randomized construction named $AD^{\log \log n}$ in [**26**] can be computed in a distributed fashion in the $\mathcal{LOCAL}$ model in polylog$n$ time and has hereditary density $\delta = O(1)$. This together with any of the previous simulators and Theorem 5.6 results in a $(O(1), \text{polylog} n)$-simulator. □

With these various simulators it is possible to simulate a computation in the $\mathcal{LOCAL}$ model with very little (polylogarithmic) multiplicative or additive overhead in the $\mathcal{GOSSIP}$ model. Note that while the complexity of the presented simulators is incomparable, one can interleave their executions (or the executions of the simulated algorithms) and thus get the best runtime for any instance. This, together with the results from Corollary 5.7 and 5.2, proves our main result of Theorem 1.2.

6. Discussion

This paper presents a more efficient alternative to the `UniformGossip` algorithm that allows fast rumor spreading on all graphs, with no dependence on their conductance. We then show how

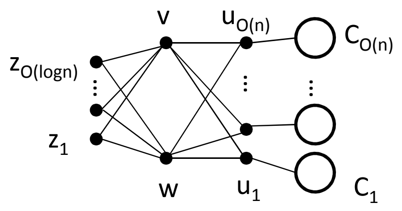

FIGURE 1. An example illustrating the behavior of the algorithm choosing a random neighbor whose information is still unknown.

this leads to fast simulation in the $\mathcal{GOSSIP}$ model of any algorithm designed for the $\mathcal{LOCAL}$ model by constructing sparse spanners. This work leaves some interesting directions for future work, which we discuss below.

First, as mentioned in the introduction, there are cases in which the algorithm where each node chooses a neighbor uniformly at random only from among those it has not yet heard from (directly or indirectly), performs slower than the optimal. An example is the graph in Figure 6, where C_i stands for a clique of size $O(1)$ in which every node is also connected to the node x_i. In this example, it takes 2 rounds for the node w to hear about the node v (through nodes in $\{z_1, \ldots, z_{O(\log n)}\}$). During these rounds there is a high probability that a constant fraction of the nodes in $\{u_1, \ldots, u_{O(n)}\}$ did not yet hear from neither v nor w. With high probability, a constant fraction of these will contact w before contacting v, after which they will not contact v anymore because they will have heard from it through w. This leaves $O(n)$ nodes which v has to contact directly (since nodes in $\{z_1, \ldots, z_{O(\log n)}\}$ are no longer active since they already heard from both of their neighbors), resulting in a linear number of rounds for NEIGHBOREXCHANGE.

We note, however, that this specific example can be solved by requiring nodes that have heard from all their neighbors to continue the algorithm after resetting their state, in the sense that they now consider all their neighbors to be such that they have not heard from (this is only for the sake of choosing the next neighbor to contact, the messages they send can include previous information they received). Therefore, we do not rule out the possibility that this algorithm works well, but our example suggests that this may not be trivial to prove.

Second, regarding our solution to the RUMOR problem, the `Superstep` algorithm, as presented, can be implemented in synchronous environments in a straightforward manner. To convert our algorithm to the asynchronous setting, one needs to synchronize the reversal step. Synchronization is a heavy-handed approach and not desirable in general.

To alleviate this problem, we believe, it is possible to get rid of the reversal step altogether. The basic idea is to do away with the hard decisions to "remove" edges once a message from a neighbor has been received. And instead to multiplicatively decrease the weight of such edges for the next round. This approach would introduce a slight asymmetry in each edge's weight in both directions. In order to analyze such an algorithm, it is needed to understand the behavior of `RandomNeighbor` in the general asymmetric setting. In this setting, each vertex uses its own distribution over outgoing links when choosing a communication partner at each step. We believe that understanding the asymmetric `RandomNeighbor` is an open problem of central importance.

References

[1] L. Barenboim and M. Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. In *Proceedings of the twenty-seventh ACM Symposium on Principles of Distributed Computing (PODC)*, pages 25–34, New York, NY, USA, 2008. ACM.

[2] L. Barenboim and M. Elkin. Distributed $(\delta+1)$-coloring in linear (in δ) time. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 111–120, New York, NY, USA, 2009. ACM.

[3] K. Censor-Hillel and H. Shachnai. Fast information spreading in graphs with large weak conductance. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 440–448, 2011.

[4] F. Chierichetti, S. Lattanzi, and A. Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 399–408, 2010.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, 1987.

[6] B. Derbel, C. Gavoille, D. Peleg, and L. Viennot. On the locality of distributed sparse spanner construction. In *Proceedings of the twenty-seventh ACM Symposium on Principles of Distributed Computing (PODC)*, pages 273–282, New York, NY, USA, 2008. ACM.

[7] B. Doerr and M. Fouz. Asymptotically optimal randomized rumor spreading. *CoRR*, abs/1011.1868, 2010.

[8] B. Doerr, T. Friedrich, and T. Sauerwald. Quasirandom rumor spreading. In *Proceedings of the nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 773–781, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[9] B. Doerr, T. Friedrich, and T. Sauerwald. Quasirandom rumor spreading: Expanders, push vs. pull, and robustness. In *36th International Colloquium on Automata, Languages and Programming (ICALP)(1)*, pages 366–377, 2009.

[10] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. *J. Comput. Syst. Sci.*, 71:467–479, November 2005.

[11] N. Fountoulakis and A. Huber. Quasirandom rumor spreading on the complete graph is as fast as randomized rumor spreading. *SIAM Journal on Discrete Mathematics*, 23(4):1964–1991, 2009.

[12] P. Fraigniaud and G. Giakkoupis. On the bit communication complexity of randomized rumor spreading. In *Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 134–143, New York, NY, USA, 2010. ACM.

[13] G. Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 57–68, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[14] G. Giakkoupis and P. Woelfel. On the randomness requirements of rumor spreading. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 449–461, 2011.

[15] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, page 565, Washington, DC, USA, 2000. IEEE Computer Society.

[16] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. In *Proceedings of the thirty-third Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–172, New York, NY, USA, 2001. ACM.

[17] D. Kempe and J. M. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 471–480, Washington, DC, USA, 2002. IEEE Computer Society.

[18] A. Kostochka. Lower bound of the hadwiger number of graphs by their average degree. *Combinatorica*, 4:307–316, 1984. 10.1007/BF02579141.

[19] F. Kuhn and T. Moscibroda. Distributed approximation of capacitated dominating sets. *Theor. Comp. Sys.*, 47:811–836, November 2010.

[20] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *Proceedings of the seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 980–989, New York, NY, USA, 2006. ACM.

[21] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986.

[22] W. Mader. Homomorphieeigenschaften und mittlere kantendichte von graphen. *Mathematische Annalen*, 174:265–268, 1967. 10.1007/BF01364272.

[23] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proceedings of the twenty-fifth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 113–122, New York, NY, USA, 2006. ACM.
[24] D. Peleg. *Distributed computing: a locality-sensitive approach.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
[25] D. Peleg and A. Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989.
[26] S. Pettie. Low distortion spanners. *ACM Transactions on Algorithms (TALG)*, 6:7:1–7:22, December 2009.
[27] S. Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Computing*, 22(3):147–166, 2010.
[28] J. Schneider and R. Wattenhofer. A new technique for distributed symmetry breaking. In *Proceeding of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 257–266, New York, NY, USA, 2010. ACM.
[29] A. Sinclair. *Algorithms for random generation and counting: a Markov chain approach.* Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993.
[30] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *CoRR*, abs/0808.4134, 2008.

Appendix A. Proof of Lemma 3.1

Proof. Assume, towards a contradiction, that $\Phi(V-S) < \xi/3$. Then, there exists a cut (P,Q) of $V-S$ with $\varphi(P,Q) < \xi/3$ and specifically

$$\max\left\{\frac{w(P,Q)}{\mathrm{vol}(P)}, \frac{w(P,Q)}{\mathrm{vol}(Q)}\right\} \le \frac{\xi}{3} \tag{3}$$

Henceforth, let Q be the smaller of the two, i.e. $\mathrm{vol}(Q) \le \mathrm{vol}(V-S)/2$.

We are going to show that $\varphi(S \cup Q, P) \le \xi$ and either $S \cup Q$ or P should have been chosen instead of S.

Consider the case $\mathrm{vol}(S \cup Q) \le \mathrm{vol}(V)/2$. In this case,

$$\varphi(S\cup Q, P) = \frac{w(S,P)+w(Q,P)}{\mathrm{vol}(S\cup Q)} = \frac{w(S,P)+w(Q,P)}{\mathrm{vol}(S)+\mathrm{vol}(Q)}$$
$$\le \max\left\{\frac{w(S,P)}{\mathrm{vol}(S)}, \frac{w(Q,P)}{\mathrm{vol}(Q)}\right\} \le \max\left\{\frac{w(S,P)+w(S,Q)}{\mathrm{vol}(S)}, \frac{w(Q,P)}{\mathrm{vol}(Q)}\right\} \le \max\left\{\xi, \xi/3\right\} = \xi$$

This establishes a contradiction, because $\varphi(S \cup Q, P) \le \xi$ and $\mathrm{vol}(S) < \mathrm{vol}(S \cup Q) \le \mathrm{vol}(V)/2$.

Now let's consider the case $\mathrm{vol}(S \cup Q) > \mathrm{vol}(V)/2$. First, we argue that $\mathrm{vol}(S \cup Q)$ cannot be too large. We use that $\mathrm{vol}(Q) \le \frac{1}{2}\mathrm{vol}(V-S) = \frac{1}{2}(\mathrm{vol}(V) - \mathrm{vol}(S))$.

$$\mathrm{vol}(S\cup Q) = \mathrm{vol}(S) + \mathrm{vol}(Q) \le \mathrm{vol}(S) + \frac{\mathrm{vol}(V)-\mathrm{vol}(S)}{2} = \frac{\mathrm{vol}(V)+\mathrm{vol}(S)}{2} \le \frac{5}{8}\mathrm{vol}(V) \tag{4}$$

Hence, $\mathrm{vol}(P) \ge \frac{3}{8}\mathrm{vol}(V)$. In addition, for the cut size, we have

$$\begin{aligned} w(S\cup Q, P) &= w(S,P) + w(Q,P) \\ &\le \xi\,\mathrm{vol}(S) + \frac{\xi}{3}\mathrm{vol}(Q) \\ &\le \xi\,\mathrm{vol}(S) + \frac{\xi}{3}\frac{\mathrm{vol}(V)-\mathrm{vol}(S)}{2} \\ &\le \frac{5}{6}\xi\,\mathrm{vol}(S) + \frac{1}{6}\xi\,\mathrm{vol}(V) \\ &= \frac{3}{8}\xi\,\mathrm{vol}(V) \end{aligned}$$

And now we can bound the cut conductance:

$$\varphi(S \cup Q, P) = \frac{w(S \cup Q, P)}{\operatorname{vol}(P)} \leq \frac{\frac{3}{8}\xi \operatorname{vol}(V)}{\frac{3}{8}\operatorname{vol}(V)} = \xi \tag{5}$$

This also establishes a contradiction because $\varphi(S \cup Q, P) \leq \xi$ while $\operatorname{vol}(S) \leq \frac{1}{4}\operatorname{vol}(V) < \frac{3}{8}\operatorname{vol}(V) \leq \operatorname{vol}(P) \leq \frac{1}{2}\operatorname{vol}(V)$. □

Appendix B. Proof of Theorem 3.3

Proof. The depth K of the recursion is, by construction, at most $\log_{4/3}\operatorname{vol}(V)$ assuming that the smallest non-zero weight is 1. Let $\mathcal{R}_i \subseteq 2^V$ be a collection of the U-parameters of invocations of `Cluster` at depth $0 \leq i \leq K$ of the recursion. (So, for example, $\mathcal{R}_0 = \{V\}$.) For a set U let $S(U)$ be the small side of the cut produced by `Cluster(`G`,`U`,`ξ`)`, or $\emptyset$ if no eligible cut was found. We can then bound the total weight of cut edges as

$$\begin{aligned}\sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i} w\big(S(U), U - S(U)\big) &\leq \sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i} \xi \operatorname{vol}\big(S(U)\big) \leq \sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i} \frac{\xi}{2}\operatorname{vol}(U)\\ &\leq \frac{\xi}{2}\sum_{0\leq i\leq K}\sum_{U\in\mathcal{R}_i}\operatorname{vol}(U) \leq \frac{\xi}{2}\sum_{0\leq i\leq K}\operatorname{vol}(V) \leq \frac{\xi\log_{4/3}\operatorname{vol}(V)}{2}\operatorname{vol}(V),\end{aligned}$$

Where we use the convention $w(\emptyset, S) = 0$. If we set $\xi = \frac{3\zeta}{\log_{4/3}\operatorname{vol}(V)}$, for some $0 < \zeta < 1$, then Corollary 3.2 establishes the theorem. □