Finding a Bounded-Degree Expander Inside a Dense One

Luca Becchetti
Sapienza Università di Roma
Rome, Italy
becchetti@dis.uniroma1.it

Andrea Clementi
Università di Roma Tor Vergata
Rome, Italy
clementi@mat.uniroma2.it

Emanuele Natale
Max Planck Institute for Informatics
Saarbrücken, Germany
emanuele.natale@mpi-inf.mpg.de

Francesco Pasquale
Università di Roma Tor Vergata
Rome, Italy
pasquale@mat.uniroma2.it

Luca Trevisan
U.C. Berkeley
Berkeley, CA, United States
luca@berkeley.edu

Abstract

It follows from the Marcus-Spielman-Srivastava proof of the Kadison-Singer conjecture that if $G = (V, E)$ is a Δ-regular dense expander then there is an edge-induced subgraph $H = (V, E_H)$ of G of constant maximum degree which is also an expander. As with other consequences of the MSS theorem, it is not clear how one would explicitly construct such a subgraph.

We show that such a subgraph (although with quantitatively weaker expansion and near-regularity properties than those predicted by MSS) can be constructed with high probability in linear time, via a simple algorithm. Our algorithm allows a distributed implementation that runs in $\mathcal{O}(\log n)$ rounds and does $\mathcal{O}(n)$ total work with high probability.

The analysis of the algorithm is complicated by the complex dependencies that arise between edges and between choices made in different rounds. We sidestep these difficulties by following the combinatorial approach of counting the number of possible random choices of the algorithm which lead to failure. We do so by a compression argument showing that such random choices can be encoded with a non-trivial compression.

Our algorithm bears some similarity to the way agents construct a communication graph in a peer-to-peer network, and, in the bipartite case, to the way agents select servers in blockchain protocols.

1 Introduction

The proof of the Kadison-Singer conjecture by Marcus, Spielman and Srivastava [13] (which we will henceforth call the *MSS Theorem*) has several important graph theoretic corollaries. In particular, if $G = (V, E)$ is an undirected graph with n nodes in which every edge has effective resistance $\mathcal{O}(n/|E|)$, then there is an edge-induced subgraph $H = (V, E_H)$ of G that has $\mathcal{O}(n/\varepsilon^2)$ edges and that is an unweighted ε-spectral-sparsifier[1] of G.

Interesting examples of graphs to which this statement applies are edge-transitive graphs, such as the hypercube, and regular expanders of constant normalized edge expansion. As with other consequences of the MSS Theorem, and other non-constructive results proved with similar techniques, it is not known how to construct such subgraphs in polynomial (or even subexponential) time.

In the case of regular expanders, the result, qualitatively, states that if $G = (V, E)$ is a Δ-regular graph of constant normalized edge expansion, there exists an edge-induced subgraph H of G that has constant maximum degree and constant normalized edge expansion.

In this work, we show how to constructively find such an H, assuming that $\Delta = \Omega(n)$ and that the second eigenvalue of the adjacency matrix of G (which measures the spectral expansion of the graph) is at most a sufficiently small constant times the degree Δ.

The randomized algorithm we propose receives as input a Δ-regular graph G and two integer parameters d and c, and it returns a subgraph H of G, in which each node has degree between d and $(c+1) \cdot d$. To this purpose, our algorithm first constructs a directed graph, in which each directed edge corresponds to an edge of G, and in which each node has outdegree d and indegree between 0 and $c \cdot d$. It then obtains the graph H by ignoring the directions of the edges. If $c > 2n/|E|$ and d is a sufficiently large absolute constant, our algorithm succeeds in identifying a subgraph H with the above properties with high probability in $\mathcal{O}(n)$ time.

Our algorithm naturally lends itself to a distributed implementation, in a model in which the underlying communication network is G itself, with its nodes as computing elements. In this model, the nodes of G can collectively identify a subgraph H with the properties mentioned above in $\mathcal{O}(\log n)$ rounds and with $\mathcal{O}(n)$ total work and communication cost, in the sense that at the end of the protocol, each node knows its neighbors in H.

If we further assume that the second eigenvalue of the adjacency matrix of G is at most $\gamma\Delta$, with γ a sufficiently small constant, we can prove that, with high probability, H has conductance $\Omega(1)$ (see Theorems 4 and 5 for formal statements of the above results).

Our algorithm RAES, (for *Request a link, then Accept if Enough Space*) is best described as a distributed protocol that works in rounds, each consisting of two phases. Initially, each node has 0 outgoing links and 0 incoming links. In the first phase of each round, each node v selects enough random neighbors (according to the topology of G) so that linking to all of them would secure v a total of d outgoing links. It then submits a request to each selected neighbor to establish a link. In the second phase of the round, each node accepts all requests received in the first phase of the current round, unless doing so would cause it to exceed the limit of cd incoming links; if this is the case, the node rejects all requests it received in the first phase. The algorithm completes when each node has established exactly d outgoing links, so that no further requests are submitted. A

[1] A weighted graph $H = (V, E_H)$ is an ε-spectral-sparsifier [2] of a graph $G = (V, E_G)$ if, for every vector $\mathbf{x} \in \mathbb{R}^V$, we have

$$(1-\varepsilon) \sum_{(u,v)\in E_G} (x_u - x_v)^2 \leqslant \sum_{(u,v)\in E_H} w_H(u,v) \cdot (x_u - x_v)^2 \leqslant (1+\varepsilon) \sum_{(u,v)\in E_G} (x_u - x_v)^2$$

where $w_H(u,v)$ is the weight of the edge (u,v) in H. We say that H is *unweighted* if the weights of all the edges of H are all equal to the same scaling factor $|E_G|/|E_H|$.

formal description of the algorithm is given in Section 2.

To show that our algorithm terminates in $\mathcal{O}(\log n)$ rounds with high probability when G is Δ-regular and $c > 2n/\Delta$, we can see that, for each round t, for each node v, for each of the requests (if any) made by v at round t, and for each fixing of all remaining randomness of the algorithm, there is at least a probability $1/2$ that the request is accepted (because, at each round, the number of nodes that reject any request is at most $n/2$). This is enough to see that convergence takes $\mathcal{O}(\log n)$ rounds with high probability and total work $\mathcal{O}(dn)$ on average.

To see that the total work is $\mathcal{O}(dn)$ with high probability, we show that, at each round, the expected number of "missing edges," that is, the number $d \cdot n - \sum_v \mathrm{d}_{\mathrm{v}}^{\mathrm{out}}$ (with $\mathrm{d}_{\mathrm{v}}^{\mathrm{out}}$ the current number of v's outgoing links), shrinks, on average, by a constant factor. Furthermore, the amount by which the above quantity changes at each step is a Lipschitz function of independent random variables, which means that we can argue with high probability about the amount by which this quantity decreases.

The main result of this work is the proof that, if G is a sufficiently good expander, then the graph produced by the algorithm has constant expansion.

In the spirit of how one analyzes the expansion of random regular graphs, we would like to argue that, for every set $S \subseteq V$ of $s \leqslant n/2$ vertices, there is at least a probability, say, $1 - n^{-2} \cdot \binom{n}{s}^{-1}$, that, of the ds outgoing links from the vertices of S, at least $\Omega(ds)$ are links from S to $V - S$. Then we could use a union bound over all possible sets S to say that with probability at least $1 - 1/n$ every set S has at least $\Omega(ds)$ links crossing the cut and going into $V - S$. The probability distribution of the links created by the algorithm, and the ways in which the links are correlated, are however very difficult to analyze.

Our approach is to use a *compression argument*: we show that the random choices of the algorithm that lead to a non-expanding graph can be non-trivially compressed, and hence have low probability. The approach of proving that an event is unlikely by showing that the random choices leading to it are compressible is often a convenient one to analyze the outcome of an algorithm. Such arguments are sometimes expressed in the language of *Kolmogorov complexity* [22] and they are often used in cryptography to analyze the security of protocols that involve a random oracle, following [6]. In [16], the authors review various probabilistic analyses that can be performed using compression argument (which they call *encoding* arguments).

Our argument is roughly as follows: suppose that, in the graph constructed by the algorithm, S is a non-expanding set of vertices. If G is a sufficiently good Δ-regular expander, then, from the expander mixing lemma, we get that the typical vertex of S has only about $\Delta \cdot |S|/n$ neighbors in S, but, if S is non-expanding in H, then the typical node in S has, say, at least $.9 \cdot d$ of its d outgoing links in S. This means that, for the typical node in S, we can represent $.9d$ of its d outgoing links using $\log \frac{\Delta \cdot |S|}{n}$ bits instead of $\log \Delta$, with a saving of order of $d|S| \log \frac{n}{|S|}$ bits. For sufficiently large constant d, this is more than the $\log \binom{n}{|S|}$ bits that it takes to represent the set S. Unfortunately, things are not so easy because we need the representations of choices made by the nodes in the algorithm to be prefix-free, in order for their concatenation to be decodable, and so we have to spend some additional bits in the representation of the choices that lead to links from S to $V - S$ (which are not so many since S is a non-expanding set) and for the choices that are rejected. To complete the argument, we have to argue that the overall number of choices of nodes of S that are rejected cannot be too large, and otherwise we would have a non-trivial way of compressing the rejected choices. This is true because, as argued above, each choice has a small probability of being rejected, and so random choices of the algorithm that lead to a lot of rejected choices are unlikely, hence compressible.

Algorithm RAES is inspired by the way nodes create bounded-degree overlay networks in real-

life distributed protocols, such as in peer-to-peer protocols [7, 17] like BitTorrent or in distributed ledger protocols such as Bitcoin [18]. In this protocol for example, each node in a communication network is aware of the existence of a certain subset of the other nodes (in our algorithm, for some node v this subset corresponds to the set of neighbors of v in G). Each node tries to establish a minimum number of connections to other nodes (or to special "server" nodes[2]) and does so by selecting them at random from its list of known nodes (or known servers). Nodes also have a maximum number of connections they are going to accept, rejecting further connections once this limit is reached.

On the other hand, our algorithm does not capture important traits of peer-to-peer and blockchain models, such as the fact that nodes can join or leave the network, and that nodes can exchange their lists of known nodes, so that the graph "G" in fact is dynamic. We believe, however, that our analysis addresses important aspects, such the complicated dependencies that arise between different links in the virtual network, and the *expansion* properties of the resulting virtual network. Expansion in particular is closely related to resilience to nodes leaving the network, a very important property in practice. We next discuss some related work.

Distributed constructions of expanders. Our main result gives a distributed algorithm to construct a bounded-degree expander. This is a question that has been addressed in a number of models and initial conditions. In [10], Law and Siu provide a distributed protocol running on the LOCAL asynchronous model that form expander graphs of arbitrary fixed degree d. Their goal is to maintain the expansion property under insertions, starting from a constant-size graphs, and they show how to do so in constant time and constant message complexity per node insertion. See also [7] and [19] for such sequential constructions of expanders.

In [1], Allen-Zhu et al. show a simple and local protocol that, starting from any connected d-regular connected graph with $d = \Omega(\log n)$, returns a d-regular expander. At every every round, an edge e is selected u.i.r. together with one length-3 path including e and, then, a suitable flipping of the edges of this path is performed (so, the obtained graph is not guaranteed to be a subgraph of the original graph). Their spectral analysis of the evolving graph shows that, after $\mathcal{O}(n^2 d^2 \text{polylog}\,(n)))$ rounds, the obtained random graphs is an expander, with high probability. Their algorithm models the way in which nodes exchange neighborhood information in real-life protocols, and it works starting from much more limited information than ours (their initial information is an arbitrary graph of logarithmic degree, while we start from a graph of linear degree which is already an expander), and the prize that they pay is a polynomial, rather than logarithmic, convergence time.

Sparsification. We motivated our main result as a constructive proof of a special case of the sparsification results implied by the MSS theorem, for which no constructive proofs are known. Here it matters that we are interested in sparsifying a regular graph by using an unweighted subgraph of bounded maximum degree. If we allowed weighted graphs, and we were only concerned about the average degree of the sparsifier, then an explicit construction of constant average-degree sparsifiers for all graphs is given by the BSS sparsifiers of [2]. A parallel construction of the BSS sparsifier, however, is not known. Parallel construction of (weighted, unbounded max degree) sparsifiers have been studied [9], but such constructions involve graphs of logarithmic average degree, a setting in which our problem is trivial: given a Δ-regular graph $G = (V, E)$, if we choose each edge independently with probability order of $(\log n)/\Delta$, we get a graph that with high probability has maximum degree $\mathcal{O}(\log n)$ and, using matrix Chernoff bound, we can show that it is a spectral

[2] In this setting, we notice that if G is bipartite then H is bipartite as well.

sparsifier of G if G is such that every edge has effective resistance $\mathcal{O}(n/|E|)$, including the case of expanders and of edge-transitive graphs.

Parallel Balls-Into-Bins Processes. If the underlying network is the complete graph K_n, then RAES can be seen as a *parallel balls-into-bins* algorithm [15, 11] where there are $m = dn$ balls, each one representing an outgoing-link request which must be assigned to one of n bins, corresponding to the nodes of the network. In this interpretation, our algorithm assigns each ball into one bin so that the maximum load of the bins is at most cd, for some constant c and terminates in $\mathcal{O}(\log n)$ rounds with high probability. Several algorithms have been introduced for this problem and the best algorithms achieve constant maximum load within a constant number of rounds by using $k > 1$ random choices at every round for each ball [11]. The RAES strategy adopted by our algorithm is similar to that used in the basic version of Algorithm PARALLELTHRESHOLD analysed in [20] by Berenbrink et Al, which is in turn a parallelized version of the scheduling strategy studied in [3]. They show that the convergence time is $\mathcal{O}(n \log m)$ when $cd = d+1$, while our analysis implies that is $\mathcal{O}(\log m)$ when c is an absolute constant larger than 1. The maximum number of balls accepted by each bin, called the *threshold*, is fixed to $\lceil m/n \rceil + 1$. They show this basic version, achieving an almost tight maximum load, converges within $\mathcal{O}(n \log m)$ rounds, w.h.p. They also conjecture a tight lower bound on the convergence time.

2 Preliminaries and main result

For an undirected graph $G = (V, E)$, the *volume* of a subset of nodes $U \subseteq V$, is $vol(U) = \sum_{u \in U} d_u$. Notice that when G is Δ-regular, we have $vol(U) = \Delta|U|$.

Definition 1. *A graph $G = (V, E)$ is an ε-expander if, for every subset $U \subset V$ with $|U| \leqslant n/2$, the size of the cut $(U, V - U)$ is at least $\varepsilon \cdot vol(U)$.*

The expansion properties we derive for the subgraph returned by Algorithm RAES turn out to depend on the spectral gap of the input graph. In particular, our analysis uses the following "one-sided" version of the *Expander Mixing Lemma* [12], which establishes a connection between the second largest eigenvalue of the adjacency matrix of G and its expansion properties and also holds for bipartite graphs.

Lemma 2. *Assume $G = (V, E)$ is a Δ-regular graph and let λ be the second largest eigenvalue of G's adjacency matrix[3]. Let S be any subset of nodes. Then, the number $e(S, S)$ of edges of G with both endpoints in S is at most*

$$\frac{1}{2}\left(\frac{\Delta|S|^2}{n} + \lambda|S|\right).$$

Proof. If 1_S is the indicator vector of S, then, it holds that

$$1_S A 1_S = \sum_{v \in V} |N_v(S)| = 2 \cdot e(S, S),$$

where $N_v(S)$ is the set of v's neighbors in S and $e(S, S)$ is the number of edges with both end-points in S. Observe that the matrix $A - \Delta J/n$ (where J is the matrix having all entries set to 1) has largest eigenvalue λ, so we get

$$1_S(A - \Delta J/n)1_S \leqslant \lambda \|1_S\|^2 = \lambda \cdot |S|.$$

[3]Since G is Δ-regular, the bounds on λ we derive immediately translate into bounds on the second largest eigenvalue of G's normalized matrix.

We also notice that $1_S(\Delta J/n)1_S = \Delta|S|^2/n$. It thus follows that

$$e(S,S) \leqslant \frac{1}{2}\left(\frac{\Delta|S|^2}{n} + \lambda|S|\right).$$

□

In the next sections, we analyze the behaviour of Algorithm RAES on dense, regular expanders. The algorithm was informally described in the introduction, a more formal description is given below.

Algorithm 1 RAES(G, d, c)

```
H := empty directed graph over the node set V
while H has nodes of outdegree < d do
    PHASE 1:                                   ▷ d_v^out: current outdegree of v in H
    for each node v ∈ V do
        v ∈ V picks d − d_v^out neighbors in G uniformly at random
        v submits a connection request to each of them
    end for
    PHASE 2:                                   ▷ d_v^in: current indegree of v in H
    for each node v ∈ V do
        if v received ⩽ cd − d_v^in connection requests in the previous phase then
            v accepts all of them and the corresponding directed edges are added to H
        else
            v rejects all connection requests received in Phase 1
        end if
    end for
end while
Replace each directed edge by an undirected one
return H
```

We next define the class of almost-regular graphs RAES stabilizes on w.h.p.

Definition 3. *A graph $G = (V, E)$ is a (d, cd)-almost regular graph if the degree d_v of any node $v \in V$ is such that $d_v \in \{d, \ldots, (c+1)d\}$.*

Our main results can be formally stated as follows.

Theorem 4. *For every $d \geqslant 1$, every $0 < \alpha \leqslant 1$, and every $c = \Omega(1/\alpha)$, the time complexity of RAES(G, d, c) is $\mathcal{O}(n)$, w.h.p., for every Δ-regular graph $G = (V, E)$ with $\Delta = \alpha n$. Moreover, the algorithm can be implemented in the uniform GOSSIP distributed model[4] so that its parallel completion time is $\mathcal{O}(\log n)$ and its overall message complexity is $\mathcal{O}(n)$, w.h.p.*

Theorem 5. *A constant $\varepsilon > 0$ exists such that, for every large-enough d, every $0 < \alpha \leqslant 1$, and every $c = \Omega(1/\alpha^2)$, RAES(G, d, c) stabilizes over a $(d, (c+1)d)$-almost regular ε-expander $H = (V, A)$, w.h.p., for every Δ-regular graph $G = (V, E)$ with $\Delta = \alpha n$ and second largest eigenvalue of the adjacency matrix $\lambda \leqslant \varepsilon\alpha^2\Delta$.*[5]

[4] In this very-restrictive communication model [4, 8], at every synchronous step, each node can (only) contact a *constant* number of its neighbors, chosen u.i.r., and exchange two messages (one for each direction) with each of them.

[5] I.e., G is a sufficiently good expander. Also note that, equivalently, we are imposing that the second largest eigenvalue of G's normalized adjacency matrix be at most $\varepsilon\alpha^2$.

The proof of Theorem 4 is given in Section 3, while the proof of Theorem 5, which is our main technical contribution, is described in Section 4.

3 Proof of Theorem 4

Throughout this section, we consider a Δ-regular graph $G = (V, E)$ with $\Delta = \alpha n$ for some arbitrary constant $0 < \alpha < 1$. We analyze the execution of Algorithm RAES on input G for any constants $d \geqslant 1$ and $c > 1/\alpha$.

Recall that, according to the process defined by RAES, each node v asks for d *link requests* to its neighbors and has cd *slots* to accomodate link requests from its neighbors.

We first provide a simple proof that RAES on input $G = (V, E)$ terminates within a logarithmic number of rounds[6], w.h.p.

Lemma 6. *For every $d \geqslant 1$, every $c > 1/\alpha$, and every $\beta > 1$, RAES(G, d, c) completes the task within $\beta \log(n)/\log(\alpha c)$ rounds, with probability at least $1 - d/n^{\beta-1}$.*

Proof. Let us fix an arbitrary ordering of the nd required links and, for $i = 1, \ldots, nd$, let $X_i^{(t)}$ be the binary random variable taking value 1 if link i is settled at the end of round t and 0 otherwise.

First note that, since a link is settled at some round t if it was already settled at previous round $t-1$, it holds that

$$\mathbf{P}\left(X_i^{(t)} = 0\right) = \mathbf{P}\left(X_i^{(t)} = 0 \mid X_i^{(t-1)} = 0\right) \mathbf{P}\left(X_i^{(t-1)} = 0\right) . \tag{1}$$

Let us name $\mathbf{Y}_{-i}^{(t)} = \left(Y_1^{(t)}, \ldots, Y_{i-1}^{(t)}, Y_{i+1}^{(t)}, \ldots, Y_{nd}^{(t)}\right)$ the random vector where, for each $j \neq i$, random variable $Y_j^{(t)}$ indicates the destination node of link j at round t. Observe that for every vector $\mathbf{y}_{-i} = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_{nd}) \in V^{nd-1}$ it holds that

$$\mathbf{P}\left(X_i^{(t)} = 0 \mid X_i^{(t-1)} = 0,\, \mathbf{Y}_{-i}^{(t)} = \mathbf{y}_{-i}\right) \leqslant \frac{1}{\alpha c} . \tag{2}$$

Indeed, given any $\mathbf{y}_{-i} \in V^{nd-1}$, there are always at most $nd/(cd) = n/c$ nodes with cd or more incoming link requests. Hence, among the αn neighbors of the node asking link i, at least $(\alpha - 1/c)n$ have less than cd incoming requests. Hence, the probability that link i settles is at least $1 - 1/(\alpha c)$. Saturating over all $\mathbf{y}_{-i} \in V^{nd-1}$, from (2) we get that $\mathbf{P}\left(X_i^{(t)} = 0 \mid X_i^{(t-1)} = 0\right) \leqslant 1/(\alpha c)$ and thus from (1) we have that $\mathbf{P}\left(X_i^{(t)} = 0\right) \leqslant 1/(\alpha c)^t$. The thesis then follows from a union bound over all the nd links and from the fact that $t \geqslant \beta \log(n)/\log(\alpha c)$. □

Remark. The first proof we gave for the above lemma was based on a simple compression argument [16]. We describe it in Appendix B since it can be used by the reader as a "warm-up" for the more difficult analysis given in Section 4.

The time complexity of Algorithm RAES is asymptotically bounded by the total number of link requests produced by its execution on graph $G = (V, E)$. Lemma 6 easily implies that this number is $\mathcal{O}(dn \log n)$, w.h.p. In the next lemma we prove a tight $\mathcal{O}(nd)$ bound.

Lemma 7. *For every $d \geqslant 1$ and every $c > 2/\alpha$, the total number of link requests made by RAES(G, d, c) (and thus the time complexity) is $\Theta(dn)$, w.h.p.*

[6] Notice that the meaning of *round* here is exactly that defined in the pseudocode of RAES.

Proof. Let us fix an arbitrary ordering of the nd required links and, for $i = 1, \ldots, nd$, let $Z_i^{(t)}$ be the binary random variable taking value 1 if link i is not yet settled at the beginning of round t and 0 otherwise. The random variable indicating the total number of link requests produced by the algorithm can thus be written as

$$Z = \sum_{t=0}^{\infty} \sum_{i=1}^{nd} Z_i^{(t)} .$$

Proceeding as in the proof of Lemma 6, it is easy to see that for every $t \in \mathbb{N}$ it holds that

$$\mathbf{E}\left[\sum_{i=1}^{nd} Z_i^{(t)}\right] \leqslant \frac{nd}{(\alpha c)^t} .$$

Hence, the total expected number of link requests is $\mathbf{E}[Z] \leqslant \frac{\alpha c}{\alpha c - 1}\, nd$.

In order to prove that $Z = \mathcal{O}(nd)$ w.h.p., we first show that whenever the number of unsettled links is above $nd/\log n$, it decreses by a constant factor, w.h.p. Formally, for any $k \geqslant nd/\log n$, we derive the following inequality

$$\mathbf{P}\left(\sum_{i=1}^{nd} Z_i^{(t)} > \frac{k}{\alpha c/2} \;\middle|\; \sum_{i=1}^{nd} Z_i^{(t)} = k\right) \leqslant e^{-\frac{k}{2\alpha^2 c^4 d^2}} . \tag{3}$$

Notice that random variables $Z_i^{(t)}$ conditional on the graph formed by the links settled at the end of round $t-1$ are not independent, so we cannot use a standard Chernoff bound. However, we can use the *method of bounded differences* [5, Corollary 5.2] (see Theorem 10 in Appendix A), since the sum of the $Z_i^{(t)}$ conditional on the graph of the $nd - k$ settled links at the end of the previous round can be written as a cd-Lipschitz function of the independent k random variables indicating the link requests at round t.

In more details, we name $u^{(t-1)}$ the set of k unsettled links at the end of round $t-1$ and consider random variables $\{Y_i\}_{i \in u^{(t-1)}}$, each of them returning the node-destination index that the non-assigned link request i tries to connect to. Observe that Y_i's are mutually independent and, moreover, the sum in (3) can be written as a deterministic function of them: $\sum_{i=1}^{nd} Z_i^{(t)} = f(Y_{i_1}, \ldots, Y_{i_k})$.

Moreover, this function is cd-Lipschitz w.r.t. its arguments: If we change one of the arguments Y_i, we are moving a request i from a node v_1 to a node v_2. The largest impact this can have on $\sum_{i=1}^{nd} Z_i^{(t)}$ is that the response for each of all the link requests sent to v_2 changes. However, if this number was already larger than cd, then the moving of link request i would not have any impact. This means that, in the worst-case, at most cd link requests trying to connect to v_2 switch from assigned to non-assigned. A symmetric argument holds for the link requests trying to connect to v_1. In formulas, for all vectors of nodes $(v_{i_1}, \ldots, v_{i_j}, \ldots, v_{i_k})$ and $(v_{i_1}, \ldots, v_{i_j}^\star, \ldots, v_{i_k})$ differing only on a single entry i_j, it holds that

$$\left| f(v_{i_1}, \ldots, v_{i_j}, \ldots, v_{i_k}) - f(v_{i_1}, \ldots, v_{i_j}^\star, \ldots, v_{i_k}) \right| \leqslant 2cd .$$

Therefore, by applying Corollary 5.2 in [5] (see also Theorem 10 in Appendix A), with $\mu = M = k/(\alpha c)$ and $\beta_j = 2cd$ for all $j = 1, \ldots, k$ we get (3).

From (3) and the chain rule, it follows that, for $T = \mathcal{O}\left(\frac{\log\log n}{\log(\alpha c/2)}\right)$ rounds, the number of unassigned link requests decreases by a factor $\alpha c/2 > 1$ at each round, w.h.p., until it becomes

smaller than $nd/\log n$. These rounds thus account for $nd\sum_{t=0}^{T}\left(\frac{2}{\alpha c}\right)^t = \mathcal{O}(nd)$ connection requests, w.h.p. Then, from Lemma 6 it follows that the remaining $\frac{nd}{\log n}$ link requests are assigned within $\mathcal{O}(\log n)$ rounds, w.h.p., thus accounting for at most further $\mathcal{O}(nd)$ additional link requests, w.h.p. □

Distributed implementation

As one can easily verify from its pseudocode, Algorithm RAES is designed to work over any synchronous parallel distributed model where the nodes of the input graph $G = (V, E)$ are the local computing units which can communicate via the bidirectional links defined by the set of edges E. We remark that, at every round, each node contacts (i.e. sends link requests to) only a constant number of its neighbors. It thus follows that RAES induces a decentralized protocol that can be implemented on the communication-constrained *uniform* GOSSIP model [4, 8]. Notice that the protocol does not require any global labeling of the nodes, rather, it requires that each node knows some local labeling of its bi-directional ports.

In this setting, Lemma 6 easily implies that every node completes all of its tasks within $\Theta(\log n)$ rounds, w.h.p.

As for the communication complexity, we observe that all the point-to-point communications made by the protocol can be encoded with 1-bit messages (accept/reject the link request). Moreover, Lemma 7 implies that the overall number of links requests (and thus of exchanged messages) is w.h.p. $\Theta(dn)$, which is clearly a tight bound for this task. We have thus completed the proof of Theorem 4.

Finally, as for the termination time of the protocol, we notice that if nodes know n (or any polynomial upper bound for it), since G is regular, then they can locally derive a sufficiently good approximation of α, i.e., $\alpha' = \alpha/\text{poly}(n)$. Then, by Lemma 6, after round $T = 2\log(n)/\log(\alpha' c)$, every node can decide to stop any action (so it terminates) and it will be aware that the protocol has completed the global task, w.h.p.

4 Proof of Theorem 5

In the previous subsection, we showed that, after $T = \mathcal{O}(\log n)$ rounds, Algorithm RAES stabilizes to a subgraph $H = (V, E_H)$ of the input graph $G = (V, E)$ that turns out to be an (d, cd)-almost regular graph. In this Section, we provide the proof of Theorem 5: we indeed show that $H = (V, E_H)$ turns out to be an ε-expander, w.h.p. In the next subsection we sketch the main arguments we use in the proof. Then in the successive subsection we provide the detailed proof.

4.1 The compressed representation: Intuition

The probability distribution of the links yielded by the random process yielded by RAES, and the ways in which the links are correlated, are very difficult to analyze. In order to cope with such technical issue, we adopt a compression argument, and show that the random choices of the algorithm that lead to a non-expading graph can be non-trivially compressed, and hence have low probability.

We think of each node as having a local source of $Td\log\Delta$ random bits, and the protocol as being deterministic as a function of these local random sources. We will show that the random sources that lead the protocol to stabilize within T rounds to a non-expanding graph are non-trivially compressible, that is, they can be described using $ndT\log\Delta - \Omega(\log n)$ bits. This means that the protocol stabilizes to an expanding graph with high probability. Suppose we have an initial

randomness configuration $R \in \{0,1\}^{ndTd\log\Delta}$ that leads to a set S of size $|S| = s \leqslant n/2$ having $\leqslant \varepsilon|S|d$ outgoing edges.

Since the graph G is a good expander, the number of edges between S and $V-S$ is approximately $\Delta \cdot |S| \cdot |V-S| \cdot \frac{1}{n}$, and so the typical node in S has about $\Delta \cdot \frac{|V-S|}{n}$ neighbors in $V-S$ and $\Delta \cdot \frac{|S|}{n}$ neighbors in S.

Then the idea is that we can use $\mathcal{O}(\log s) + \log\binom{n}{s}$ bits to represent S, and then we have at least $(1-\varepsilon)sd$ edges (the outgoing edges from vertices in S that stay in S) that require only $\log\frac{\Delta|S|}{n}$ bits instead of $\log\Delta$ bits to be represented. The former costs us about $s\log\frac{n}{s}$ bits to represent, the latter involves a saving of about $(1-\varepsilon)ds\log\frac{n}{s}$ bits, so we are good provided that ε is small and d is large. (We have been making calculations as if every node in S had roughly $\Delta \cdot \frac{|S|}{n}$ G-neighbors in S, while this is only the average; a convexity argument reduces the general case to the case in which all nodes in S have the same number of G-neighbors in S.)

For this to work, however, we also need to specify which of the requests made by each node $v \in S$ are accepted and which are not. For the accepted request, we also have to specify which accepted requests are directed toward nodes in S and which are not. The latter costs us about another $\mathcal{H}(\varepsilon) \cdot d \cdot s$ bits[7], which we can afford (this is actually a non-trivial calculation that we will return to), but the former is a big problem. Naively, it would cost us $s \cdot \log\binom{dT}{d}$, which we cannot afford because we are saving only order of $d\log\frac{n}{s}$ bits per node in S, which is a constant if d and n/s are constants, while $\log\binom{dT}{d}$ grows with n for the values of T we are interested in (Lemma 6).

But we can be a bit more careful. Let ℓ_v denote the total number of link requests made by v, out of which $\ell_v - d$ are rejected and d are accepted. Then, to specify which attempted connections of v are accepted we only need $\mathcal{O}(\log\ell_v) + \log\binom{\ell_v}{d}$ bits. Note also that the typical value of ℓ_v is $(1+o_c(1)) \cdot d$, in which case the above is only $o_c(d)$ bits per node, which we can afford. Of course, it is not true that every node v has $\ell_v \leqslant (1+o_c(1)) \cdot d$, but the nodes v which have larger ℓ_v have atypical, and compressible, local random choices. The gain that we will get from this additional compression, which will be about $(\ell_v - d) \cdot \log c$, will balance the loss caused by representing which attempted connections are accepted and which ones are not.

4.2 The compressed representation: Full description

We next show that the probability that RAES on the input graph G completes within T rounds returning a graph H which is not not an ε-expander is $\mathcal{O}(n^{-\beta})$ for a constant β.

To this purpose, we proceed as follows. Assume again that $R \in \{0,1\}^{ndT\log\Delta}$ is an initial randomness configuration, corresponding to an execution terminating within T rounds with a non ε-expanding graph H. We prove that, if this is the case, we can produce a new lossless configuration R', such that R' provides the same information as R, but has length $ndT\log\Delta - \Omega(\log n)$ bits. This is enough to prove the claim.[8]

We use the following notation throughout the remainder of this proof. For a node $v \in S$, we denote by δ_v the fraction of v's edges in E that have an end-point in $V-S$, i.e., $|e(v, V-S)| = \delta_v\Delta$. We denote by ε_v the fraction of v's accepted link requests (so edges of subgraph H) with end-points in $V-S$. We further let $\delta = \frac{1}{s}\sum_{v\in S}\delta_v$ and $\varepsilon = \frac{1}{s}\sum_{v\in S}\varepsilon_v$.

To realize the compressed representation claimed above, we need the following notions.

[7] As usual, function $\mathcal{H}(x)$ denotes the binary entropy.

[8] Note that the "bad events" we are encoding are of the form "RAES stabilizes within T rounds" AND "The resulting graph H is not an ε-expander". If we consider $T = \mathcal{O}(\log n)$, together with Lemma 6 this is enough to prove Theorem 5.

Definition 8. *We say that a node w is* semi-saturated at round t *if the number of accepted incoming connection up to round $t-1$, plus the number of requested connection at round t from nodes in $V-S$ is at least $cd/2$. We say that a node w is* critical at round t *if it is not semi-saturated at round t but it has more than cd connections (accepted or requested) at round t.*[9]

We make use of the following facts.

Claim 1. *For every round t, the number of semi-saturated (resp. critical) nodes is at most $2n/c$ (resp. n/c).*

Proof. Consider a node that is semi-saturated at round t. This node was the recipient of at least $cd/2$ link requests, that it either accepted *before* round t, or it received in round t. Since, from the definition of RAES, for every node, the overall number of its link requests that are accepted within round $t-1$, plus the number of link requests it issues at round t cannot exceed d, we have a total of at most dn such requests over the entire network. This immediately implies that the number of semi-saturated nodes at round t cannot exceed $2n/c$. The argument for the number of critical nodes at round t proceeds along the same lines and is omitted for the sake of brevity. □

4.2.1 Compressed encoding and correctness

In the paragraphs that follow, we describe how the evolution of the protocol is encoded in the presence of a non-expanding subset S (with $|S| = s \leqslant n/2$ without loss of generality). We also show correctness of our encoding, discussing how the original information can be recovered from the compressed encoding, at least in those cases where this is not obvious.

Set S. We represent S by writing the number $s := |S|$ in a prefix-free way using $2\log s$ bits, and then writing the number k such that S is the k-th set of size s in lexicographic order, which takes $\log \binom{n}{s}$ bits. In total, the cost to encode S is

$$2\log s + \log \binom{n}{s}. \tag{4}$$

Requested connections originating from S that are accepted. For each node $v \in S$, we write, in a prefix free way, the number ℓ_v, using $2\log \ell_v$ bits. Then we specify, within the first ℓ_v random choices, the d ones that correspond to accepted requests.[10] The overall cost incurred is

$$\sum_{v\in S} 2\log \ell_v + \log \binom{\ell_v}{d}. \tag{5}$$

Randomness of nodes in $V-S$. We represent the randomness of nodes in $V-S$ as it is, with no gain or loss.

Subset of accepted connections from S to $V-S$. For each node $v \in S$, recall that $\varepsilon_v d$ is the number of outgoing edges from v into $V-S$. We represent the subset of accepted requested that do so, using the same encoding used above. In total, our cost is

$$\sum_{v\in S} 2\log(\varepsilon_v d) + \log \binom{d}{\varepsilon_v d}. \tag{6}$$

[9] Note that this implies that w received more than $cd/2$ requests from S at round t.

[10] Again we encode the integer i, such that the d accepted requests correspond to the i-th subset of requests of size d.

Destinations of accepted requests originating from S. For each node $v \in S$, we represent accepted connections to $V - S$ as they are (i.e., using $\log \Delta$ bits), with no gain or loss. Connections with destination in S can be represented using $\log((1-\delta_v)\Delta)$ bits instead of $\log \Delta$. Overall, the cost we incur is

$$\sum_{v \in S} (1-\varepsilon_v) d \log((1-\delta_v)\Delta) + \sum_{v \in S} \varepsilon_v d \log \Delta \,. \tag{7}$$

Note that, to represent the destinations of a generic node v's accepted requests, we would use $d \log \Delta$ bits without any compression.

Subset of semi-saturated nodes at each step. From its definition, the set of semi-saturated nodes needs not be represented explicitely. In fact, for every round t, this set is uniquely determined from the evolution of the protocol up to round $t-1$ and the requests from nodes in $V - S$ at round t, whose randomness is represented as it is.

Subset of critical nodes at each step. Let C_t be the set of critical nodes at round t and $c_t := |C_t|$. We represent all such sets. This costs us

$$\sum_{t=1}^{T} 2 \log c_t + \log \binom{n}{c_t} \,. \tag{8}$$

Destinations of rejected requests originating from S. To compress the representation of these choices, we leverage Claim 1. Specifically, for each node v, at each round t, we represent each rejected connection going to a critical node using $\log c_t$ bits, and each other rejected connection, which necessarily goes to a semi-saturated node, using $\log 2n/c$ bits. Let $rc_t(v)$ be the number of rejected connection requests from v to critical nodes in round t, and $rss(v)$ be the number of rejected connection requests from v to semi-saturated nodes, over the whole process. Overall, this costs us

$$\sum_{v \in S} \left((\ell_v - d) + \left(rss(v) \cdot \log \frac{2n}{c} \right) + \left(\sum_{t=1}^{T} rc_t(v) \cdot \log c_t \right) \right) . \tag{9}$$

Unused randomness. For every node v, we have enough randomness to describe exactly dT choices. If v completes its execution of the protocol after performing ℓ_v requests, the remaining randomness (corresponding to $dT - \ell_v$ requests) is not used. This unused randomness is both present in the uncompressed representation R and in its compressed counterpart R' and is represented as is in R', thus corresponding to $\sum_{v \in V} (dT - \ell_v) \log \Delta$ bits.

4.2.2 Compression

In this section, we show that, if R represents an execution terminating and returning a *non expanding* graph H, the corresponding encoding according to the scheme presented in the previous section uses $ndT \log \Delta - \Omega(\log n)$. In more detail, we apply our encoding scheme to any subset $S \subset V$ that is not an "ε-expander" in the graph H returned by RAES.

The proof of compression proceeds by carefully bounding the costs of the compressed representation R' and comparing them with their counterparts in the uncompressed representation R. We first show that the additive costs (with respect to R) of representing the non-expanding set S (4) and the subset of accepted requests with destinations in $V - S$ (6) are more than compensated by the compression achieved in the representation of accepted requests with destinations in S (7),

with total savings $\Omega(ds \log \frac{n}{s})$. This first step corresponds to bounding the partial cost (4) + (6) + (7).

If this is intuitively the key argument, it is neglecting the fact that we now need to identify the subset of requests originating from S that are accepted (5). For each node, this cost depends on the number of failures and in general cannot be compensated by the aforementioned savings. To this purpose, we need to exploit the further property that, for each node, failures have destinations that, for each round t, correspond to an $\mathcal{O}(1/c)$ fraction of the vertices. Compressing the destinations of these requests allows to compensate the aforementioned cost almost entirely. This second step corresponds to bounding the partial cost (5) + (8) + (9).

Bounding (4) + (6) + (7). We begin by bounding the overall compressed cost of representing the destinations of accepted requests from nodes in S, as well as the corresponding savings with respect to R. This is a crucial contribution, which depends on the expansion properties of the underlying graph G.

Claim 2. *Under the hypotheses of Theorem 5, the overall cost of representing the subset of accepted requests from nodes in S is at most*

$$\sum_{v \in S} (1-\varepsilon_v) d \log((1-\delta_v)\Delta) + \sum_{v \in S} \varepsilon_v d \log \Delta \leqslant sd \log \Delta - \frac{1-\varepsilon}{2} sd \log \frac{n}{s} + 2\varepsilon ds \,, \tag{10}$$

thus the saving is at least

$$\frac{1-\varepsilon}{2} sd \log \frac{n}{s} - 2\varepsilon ds$$

bits over the uncompressed representation.

Proof. We directly give a lower bound to the savings achieved with respect to the uncompressed representation, namely, we prove that

$$sd \log \Delta - \left(\sum_{v \in S} (1-\varepsilon_v) d \log((1-\delta_v)\Delta) + \sum_{v \in S} \varepsilon_v d \log \Delta \right) \geqslant \frac{1-\varepsilon}{2} ds \log \frac{n}{s} - 2\varepsilon ds \,,$$

whence (10) immediately follows. First of all we have

$$\begin{aligned} &sd \log \Delta - \left(\sum_{v \in S} (1-\varepsilon_v) d \log((1-\delta_v)\Delta) + \sum_{v \in S} \varepsilon_v d \log \Delta \right) \\ &= d \sum_{v \in S} (1-\varepsilon_v) \log \Delta - d \sum_{v \in S} (1-\varepsilon_v) \log((1-\delta_v)\Delta) = d \sum_{v \in S} (1-\varepsilon_v) \log \frac{1}{1-\delta_v} \,. \end{aligned}$$

Next, we consider two cases.

Case $s < \alpha\Delta$. In this case, we use the following observation, that is obviously true, since $\delta_v s$ is exactly the number of v's edges that have the other end-point in $V - S$

Observation 9.

$$1 - \delta_v \leqslant \frac{s}{\Delta} \,.$$

In this case

$$d\sum_{v\in S}(1-\varepsilon_v)\log\frac{1}{1-\delta_v}\geqslant d\sum_{v\in S}(1-\varepsilon_v)\log\frac{\Delta}{s}=(1-\varepsilon)sd\log\frac{\Delta}{s}>\frac{1-\varepsilon}{2}sd\log\frac{n}{s}\,,$$

where we used Observation 9 to write the first inequality, while the last inequality follows since in this case

$$\frac{\Delta}{s}>\frac{1}{\alpha}=\frac{n}{\Delta}\,,$$

which in turn implies

$$\left(\frac{\Delta}{s}\right)^2>\frac{n}{s}\,.$$

The case $\alpha\Delta\leqslant s\leqslant n/2$. In this case, the proof is a bit more articulated. To begin, we can write

$$\sum_{v\in S}(1-\varepsilon_v)\log\frac{1}{1-\delta_v}=-\sum_{v\in S}(1-\varepsilon_v)\log(1-\delta_v)=-(1-\varepsilon)s\sum_{v\in S}\frac{1-\varepsilon_v}{(1-\varepsilon)s}\log(1-\delta_v)$$
$$\geqslant-(1-\varepsilon)s\log\frac{\sum_{v\in S}(1-\varepsilon_v)(1-\delta_v)}{(1-\varepsilon)s}\geqslant-(1-\varepsilon)s\log\frac{\sum_{v\in S}(1-\delta_v)}{(1-\varepsilon)s}=(1-\varepsilon)s\log\frac{1-\varepsilon}{1-\delta}\,.$$

Here, to derive the third inequality we used Jensen's inequality on $\sum_{v\in S}\frac{1-\varepsilon_v}{(1-\varepsilon)s}\log(1-\delta_v)$, which is a convex combination of values of a concave function. Next, we use the following simple claim.

Claim 3.

$$1-\delta\;\leqslant\;\frac{s}{n}+\frac{\lambda}{\Delta}\,.$$

Proof. From the definition of δ_v we have

$$e(S,S)=\sum_{v\in S}(1-\delta_v)\Delta=(1-\delta)s\Delta\,.$$

The claim then follows immediately from Lemma 2. □

Next, using the above derivations and Claim 3, we can write:

$$d\sum_{v\in V}(1-\varepsilon_v)\log\frac{1}{1-\delta_v}\geqslant(1-\varepsilon)ds\log\frac{1-\varepsilon}{1-\delta}\geqslant(1-\varepsilon)ds\log\frac{(1-\varepsilon)n\Delta}{\lambda n+\Delta s}\geqslant(1-\varepsilon)sd\log\frac{n}{s}-2\varepsilon ds\,.$$

To prove the last inequality, we note that

$$\frac{(1-\varepsilon)n\Delta}{\lambda n+\Delta s}\geqslant\frac{n}{s}\cdot\frac{1-\varepsilon}{1+\varepsilon}\,,$$

whenever

$$\frac{1}{s}\leqslant(1+\varepsilon)\frac{\Delta}{s\Delta+\lambda n}\,,$$

which happens if $\lambda n\leqslant\varepsilon s\Delta$. Since we are assuming $s\geqslant\alpha\Delta$, this is certainly true if $\lambda\leqslant\varepsilon\alpha^2\Delta$. Substituting back into $\log\frac{(1-\varepsilon)n\Delta}{\lambda n+\Delta s}$ straightforwardly yields the result. □

We can now bound (4) + (6) + (7). First notice that

$$(4) = 2\log s + \log\binom{n}{s} \leqslant 3s\log\frac{n}{s}\,,$$

since we are assuming $\alpha^2 n \leqslant s \leqslant n/2$ and it holds $\log\binom{n}{s} \leqslant s\log(ne/s)$.

As for (6) we have

$$2\sum_{v\in S}\log\varepsilon_v d = 2\log\prod_{v\in S}(\varepsilon_v d) \leqslant 2\log\left(\frac{\sum_{v\in S}\varepsilon_v d}{s}\right)^s = 2s\log\varepsilon d\,,$$

where the second inequality follows from the AM-GM inequality [21], while the last equality follows from the definition of ε. We further have

$$\sum_{v\in S}\log\binom{d}{\varepsilon_v d} \leqslant d\sum_{v\in S}\varepsilon_v\log\frac{e}{\varepsilon_v} = \varepsilon ds\log e + \sum_{v\in S}\varepsilon_v\log\frac{1}{\varepsilon_v} \leqslant \varepsilon ds\log e + \varepsilon s\log\frac{1}{\varepsilon}\,.$$

Here, the second equality follows since $\sum_{v\in S}\varepsilon_v = \varepsilon s$ by definition, while the third inequality follows since the optimum of the following problem

$$\begin{aligned}&\max g(x_1,\ldots,x_k) = \sum_{i=1}^k x_i\log\frac{1}{x_i}\\ &\sum_{i=1}^k x_i = A\\ &x_i \geqslant 0\,,\end{aligned}$$

is achieved when $x_1 = \ldots = x_k = A/k$. So, overall we have

$$\begin{aligned}(4)+(6)+(7) &\leqslant 3s\log\frac{n}{s} + \underbrace{2s\log\varepsilon d + \varepsilon ds\log e + \varepsilon s\log\frac{1}{\varepsilon}}_{*} + ds\log\Delta - \frac{1-\varepsilon}{2}ds\log\frac{n}{s} + 2\varepsilon ds\\ &\leqslant 3s\log\frac{n}{s} + ds\log\Delta - \frac{1-13\varepsilon}{2}ds\log\frac{n}{s} + 2\varepsilon ds\,,\end{aligned} \tag{11}$$

where the last inequality holds, since each of the starred terms is at most $2\varepsilon ds\log\frac{n}{s}$, whenever $s \leqslant n/2$ and $d \geqslant \frac{1}{2}\log\frac{1}{\varepsilon}$.

Bounding (5) + (8) + (9). For (5) we have

$$(5) = \sum_{v\in S}\left(2\log\ell_v + \log\binom{\ell_v}{d}\right) \leqslant a(\ell_v - d) + \frac{1}{4}d\,,$$

where a is an absolute constant and where, to derive the second inequality, we used the following fact:

Claim 4. *For every b there is a sufficiently large a such that, for every d and every $\ell > d$ we have*

$$\log\binom{\ell}{d} \leqslant a\cdot(\ell - d) + bd\,.$$

As for (8), the definition of critical node at round t implies that each critical node at round t is receiving more than $cd/2$ of its incoming requests from nodes in S. As a consequence, we get

$$\sum_{v\in S} rc_t(v) > \frac{cd}{2}c_t\,,$$

For the first term in (8) this implies

$$2\sum_{t=1}^{T}\log c_t \leqslant 2\sum_{t=1}^{T}\log\left(\frac{2}{cd}\sum_{v\in S} rc_t(v)\right) < \frac{4}{cd}\sum_{t=1}^{T}\sum_{v\in S} rc_t(v) \leqslant \frac{4}{cd}\sum_{v\in S}(\ell_v - d)\,,$$

where to derive the last inequality we exchanged the order of summation and used the fact that $\sum_{t=1}^{T} rc_t(v) \leqslant \ell_v - d$, while for the for the second term in (8) we have

$$\log\binom{n}{c_t} \leqslant c_t \log\frac{en}{c_t} \leqslant \frac{2}{cd}\sum_{v\in S} rc_t(v)\log\frac{en}{c_t} \leqslant \sum_{v\in S} rc_t(v)\log\frac{2n}{c\cdot c_t}.$$

Here, to derive the third inequality we used the following

Claim 5.

$$\log\left(\frac{en}{c_t}\right)^{\frac{2}{cd}} \leqslant \log\frac{2n}{c\cdot c_t}\,,$$

whenever c is large enough that $d \geqslant \frac{2}{c}\log\frac{ce}{2}$.

Proof. The proof follows from simple calculus, recalling that the definition of critical node implies that there are at most n/c of them at any round t. □

At this point, we give an upper bound to (5) + (8) + (9). Let $\gamma = 1 + a + \frac{4}{cd}$. We have

$$\begin{aligned}
(5)+(8)+(9) &\leqslant \gamma\sum_{v\in S}\left((\ell_v - d) + \frac{1}{4}d + \sum_{t=1}^{T} rc_t(v)\log\frac{2n}{c\cdot c_t} + rss(v)\cdot\log\frac{2n}{c} + \sum_{t=1}^{T} rc_t(v)\cdot\log c_t\right)\\
&= \gamma\sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds + \sum_{v\in S}\left(rss(v)\cdot\log\frac{2n}{c} + \sum_{t=1}^{T} rc_t(v)\log\frac{2n}{c}\right)\\
&\leqslant \gamma\sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds + \log\Delta\sum_{v\in S}(\ell_v - d) - \frac{1}{2}\sum_{v\in S}(\ell_v - d)\log c\\
&\leqslant \log\Delta\sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds\,. \qquad (12)
\end{aligned}$$

Here, in the third inequality we used $\Delta = \alpha n$ and we considered $c \geqslant (2/\alpha)^2$, so that $\Delta \geqslant 2n/\sqrt{c}$, while in the last inequality we used the fact that $\gamma \leqslant \log\frac{c}{2}$ for sufficiently large c. As a result, the total number of bits to encode the executions of nodes in S is as follows (recall that we use $(n-s)dT\log\Delta$ bits for vertices in $V - S$).

$$(4)+(6)+(7)+(5)+(8)+(9) \leqslant 3s\log\frac{n}{s} + ds\log\Delta - \frac{1-13\varepsilon}{2}ds\log\frac{n}{s} + 2\varepsilon ds + \log\Delta\sum_{v\in S}(\ell_v - d) + \frac{1}{4}ds\,. \qquad (13)$$

To this cost, we should also add the randomness that was not used, which is exactly $(dT - \ell_v)\log\Delta$ for the generic node v. Our savings are then

$$\text{Savings} \geqslant dsT\log\Delta - \left(3s\log\frac{n}{s} + ds\log\Delta - \frac{1-13\varepsilon}{2}ds\log\frac{n}{s} + 2\varepsilon ds + \log\Delta\sum_{v\in S}(dT-d) + \frac{1}{4}ds\right)$$
$$= -3s\log\frac{n}{s} + \frac{1-13\varepsilon}{2}ds\log\frac{n}{s} - \left(\frac{1}{4} + 2\varepsilon\right)ds\,,$$

Finally, the expression above is $\Omega(\log n)$, as soon as ε is a sufficiently small (absolute) constant.[11]

5 Future work

A first interesting open problem is extending the analysis of RAES to non-dense expanders, i.e., to cases in which $\Delta = o(n)$. In this setting, both the proofs of convergence and of expansion might need to be revisited in significant ways. For example, if $\Delta < n/c$, we can no longer guarantee that all nodes will eventually establish d connections: it might well be the case that all neighbours of some node become saturated at some point, before the node itself can see all its requests accommodated. In fact, $\Delta = \Omega(\log n)$ is necessary to ensure that this does not occur with high probability. Another interesting generalization is extending the analysis to the case of non-regular graphs, possibly relying on the corresponding generalization of the Expander Mixing Lemma. Finally, it would be interesting to investigate the robustness of RAES in dynamic settings, in which nodes and/or vertices of the underlying graph G may join or leave the network.

References

[1] Zeyuan Allen-Zhu, Aditya Bhaskara, Silvio Lattanzi, Vahab Mirrokni, and Lorenzo Orecchia. Expanders via local edge flips. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 259–269. Society for Industrial and Applied Mathematics, 2016. 3

[2] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012. 1, 3

[3] Petra Berenbrink, Kamyar Khodamoradi, Thomas Sauerwald, and Alexandre Stauffer. Balls-into-bins with nearly optimal load distribution. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pages 326–335, New York, NY, USA, 2013. ACM. 4

[4] Keren Censor-Hillel, Bernhard Haeupler, Jonathan A. Kelner, and Petar Maymounkov. Global computation in a poorly connected world: fast rumor spreading with no dependence on conductance. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 961–970, 2012. 5, 8

[5] D.P. Dubhashi and A. Panconesi. *Dubhashi, Devdatt P., and Alessandro Panconesi. Concentration of measure for the analysis of randomized algorithms.* Cambridge University Press, 2009. 7, 19

[11] E.g., $\varepsilon \leqslant 1/51$ will suffice.

[6] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 305–313. IEEE, 2000. 2

[7] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: Algorithms and evaluation. *Perform. Eval.*, 63(3):241–263, March 2006. 3

[8] Bernhard Haeupler. Simple, fast and deterministic gossip and rumor spreading. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 705–716, 2013. 5, 8

[9] Ioannis Koutis and Shen Chen Xu. Simple parallel and distributed algorithms for spectral graph sparsification. *TOPC*, 3(2):14:1–14:14, 2016. 3

[10] Ching Law and K-Y Siu. Distributed construction of random expander networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2133–2143. IEEE, 2003. 3

[11] Christoph Lenzen and Roger Wattenhofer. Tight bounds for parallel randomized load balancing: Extended abstract. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 11–20, New York, NY, USA, 2011. ACM. 4

[12] D.A. Levin and Y. Peres. *Markov Chains and Mixing Times: Second Edition*. MBK. American Mathematical Society, 2017. 4

[13] Adam W Marcus, Daniel A Spielman, and Nikhil Srivastava. Interlacing families ii: Mixed characteristic polynomials and the Kadison–Singer problem. *Annals of Mathematics*, pages 327–350, 2015. 1

[14] Colin McDiarmid. Concentration. in probabilistic methods for algorithmic discrete mathematics, mcdiarmid c., ramirez-alfonsin j., reed b. (eds). *Algorithms and Combinatorics, Springer, Berlin, Heidelberg*, 16:195–248, 1998. 19

[15] Adler Micah, Chakrabarti Soumen, and Rasmussen Lars E. Parallel randomized load balancing. *Random Struct. Algorithms*, 13(2):159–188, 1998. 4

[16] Pat Morin, Wolfgang Mulzer, and Tommy Reddad. Encoding arguments. *ACM Comput. Surv.*, 50(3):46:1–46:36, July 2017. 2, 6, 19

[17] Thomas Moscibroda, Stefan Schmid, and Rogert Wattenhofer. On the topologies formed by selfish peers. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 133–142, New York, NY, USA, 2006. ACM. 3

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. White Paper, 2008. 3

[19] Moni Naor and Udi Wieder. Novel architectures for p2p applications: The continuous-discrete approach. *ACM Trans. Algorithms*, 3(3), August 2007. 3

[20] Berenbrink Petra, Friedetzy Tom, Lammersen Christiane, and Sauwervald Thomas. Parallel randomized load balancing. *Unpublished Manuscript*. 4

[21] J Michael Steele. *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press, 2004. 14

[22] Paul MB Vitanyi and Ming Li. *An introduction to Kolmogorov complexity and its applications*, volume 34. Springer Heidelberg, 1997. 2

Appendix

A Mathematical Tools

In Section 3, we used the method of bounded differences. In particular, we applied the following concentration bound [5, 14].

Theorem 10. *Let $\mathbf{Y} = (Y_1, \ldots, Y_m)$ be independent r.v.s, with Y_j taking values in a set A_j. Suppose the real-valued funcion f defined on $\prod A_j$ satisfies*

$$|f(\mathbf{y}) - f(\mathbf{y}')| \leqslant \beta_j$$

whenever vectors $\mathbf{y}$ and $\mathbf{y}'$ differs only in the j-th coordinate. Let μ be the expected value of r.v. $F(\mathbf{Y})$. Then, for any $M > 0$, it holds that

$$\Pr\left(F(\mathbf{Y}) - \mu \geqslant M\right) \leqslant e^{-\frac{2M^2}{\sum_{j=1}^{m} \beta_j^2}}.$$

B Proving Lemma 6 via an encoding argument

We provide here an elegant, alternative proof for the fact RAES on graph G completes its task within a logarithmic number of rounds, w.h.p. The proof relies on a simple encoding argument [16] and it can be seen as a "warm-up" for the much more complex analysis given in Section 4 which makes use of the same approach.

Lemma 11. *Let $G = (V, E)$ be any Δ-regular graph with $\Delta = \alpha n$ and $0 < \alpha \leqslant 1$ and let $d \geqslant 1$ be any absolute constant. Then, for any $c > 1/\alpha$, any $\beta > 2$, and any large-enough n, RAES(G, d, c) on graph G terminates within $(\beta/\log(\alpha c)) \log n$ rounds with probability at least $1 - n^{-(\beta-2)/2}$.*

Proof. We prove the lemma via an *encoding* argument [16]. Notice that any execution of RAES for t rounds is completely determined by a sequence of $tnd \log \Delta$ random bits: Indeed, $\log \Delta$ random bits can be used for each link request of any fixed node, each node makes at most d link requests at each round, and this procedure is repeated for every node and for t rounds.

Consider an execution where there is a node v with $d' < d$ outgoing edges at round t and note that this node must have had at least one rejected request in each of the t rounds. We can encode the sequence of $tnd \log \Delta$ bits that generates such an execution as follows: The first $\log n$ bits encode node v; The following $t(n-1)d \log \Delta$ bits encode all possible random choices of all nodes but v; As for the random bits of node v, let ℓ_v be the number of random choices actually made by v during the t rounds; we can use

- $2 \log \ell_v$ bits to encode in a prefix-free way the number ℓ_v;
- $2 \log d'$ bits to encode in a prefix-free way the number d' of accepted requests;
- $\log \binom{\ell_v}{d'}$ bits to encode the positions of the accepted requests in the sequence of ℓ_v requests;
- $\log \Delta$ bits for each one of the d' accepted request and $\log(n/c)$ bits for each one of the $\ell_v - d'$ rejected ones. Notice that we can use only $\log(n/c)$ bits instead of $\log \Delta$ to encode a rejected request since (i) each rejected request was directed to a node that was *overloaded* (i.e., a node that received at least further cd incoming requests) at the time of the request from v; (ii) since each node sends at most d requests at each round, it follows that the number of overloaded nodes is at most n/c at each round; (iii) the previous bits of our encoding uniquely identify the set of overloaded nodes at each round.

In contrast to the $\ell_v \log \Delta$ bits used by node v in the uncompresed encoding we thus use only

$$\begin{aligned}
&2\log \ell_v + 2\log d' + \log \binom{\ell_v}{d'} + d' \log n + (\ell_v - d') \log(n/c) = \\
&= \ell_v \log(n/c) + 2\log(\ell_v d') + d' \log c + \log \binom{\ell_v}{d'} = \\
&= \ell_v \log \Delta - \left[\ell_v \log(\alpha c) - 2\log(\ell_v d') - d' \log c - \log \binom{\ell_v}{d'} \right]
\end{aligned}$$

Hence, the total number of bits for node v saved in our encoding is

$$\begin{aligned}
&\ell_v \log(\alpha c) - d' \log c - \log \binom{\ell_v}{d'} - 2\log(\ell_v d') \\
&\geqslant \ell_v \log(\alpha c) - d' \log c - d' \log \frac{e\ell_v}{d'} - 2\log(\ell_v d') \\
&\geqslant (1/2)\ell_v \log(\alpha c)
\end{aligned}$$

where in the first inequality we used that $\log \binom{\ell_v}{d'} \leqslant d' \log(e\ell_v/d')$ and the last inequality holds for ℓ_v large enough, since c, d', and α are $\mathcal{O}(1)$.

By using that $\ell_v \geqslant t \geqslant (\beta/\log(\alpha c)) \log n$ and that in our encoding we use $\log n$ bits to identify node v, the fraction of strings determining an execution such that there is a node v with $d' < d$ outgoing edges at round t is thus at most

$$2^{-(1/2)\ell_v \log(\alpha c) + \log n} \leqslant 2^{-(1/2)t \log(\alpha c) + \log n} \leqslant 2^{-(1/2)\beta \log n + \log n} = n^{-(\beta-2)/2}\,.$$

□