# Expander Graph and Communication-Efficient Decentralized Optimization

Yat-Tin Chow[1], Wei Shi[2], Tianyu Wu[1] and Wotao Yin[1]

[1]Department of Mathematics, University of California, Los Angeles, CA
[2] Department of Electrical & Computer Engineering, Boston University, Boston, MA

***Abstract*—In this paper, we discuss how to design the graph topology to reduce the communication complexity of certain algorithms for decentralized optimization. Our goal is to minimize the total communication needed to achieve a prescribed accuracy. We discover that the so-called *expander graphs* are near-optimal choices.**

**We propose three approaches to construct expander graphs for different numbers of nodes and node degrees. Our numerical results show that the performance of decentralized optimization is significantly better on expander graphs than other regular graphs.**

***Index Terms*—decentralized optimization, expander graphs, Ramanujan graphs, communication efficiency.**

## I. Introduction

### A. Problem and background

Consider the decentralized consensus problem:

$$\underset{x\in\mathbb{R}^P}{\operatorname{minimize}}\, f(x) = \frac{1}{M}\sum_{i=1}^{M} f_i(x)\,, \tag{1}$$

which is defined on a connected network of $M$ agents who cooperatively solve the problem for a common minimizer $x \in \mathbb{R}^P$. Each agent $i$ keeps its private convex function $f_i : \mathbb{R}^P \to \mathbb{R}$. We consider synchronized optimization methods, where all the agents carry out their iterations at the same set of times.

Decentralized algorithms are applied to solve problem (1) when the data are collected or stored in a distributed network and a fusion center is either infeasible or uneconomical. Therefore, the agents in the network must perform local computing and communicate with one another. Applications of decentralized computing are widely found in sensor network information processing, multiple-agent control, coordination, distributed machine learning, among others. Some important works include decentralized averaging [1], [2], [3], learning [4], [5], [6], estimation [7], [8], [9], [10], sparse optimization [11], [12], and low-rank matrix completion [13] problems.

In this paper, we focus on the communication efficiency of certain decentralized algorithms. Communication is often the bottleneck of distributed algorithms since it can cost more time or energy than computation, so reducing communication is a primary concern in distributed computing.

Often, there is a trade-off between *communication requirement* and *convergence rates* of a decentralized algorithm. Directly connecting more pairs of agents, for example, generates more communication at each iteration but also tends to make faster progress toward the solution. To reduce the *total* communication for reaching a solution accuracy, therefore, we must find the balance. In this work, we argue that this balance is approximately reached by the so-called Ramanujan graphs, which is a type of expander graph.

### B. Problem reformulation

We first reformulate problem (1) to include the graph topological information. Consider an undirected graph $G = (V, E)$ that represents the network topology, where $V$ is the set of vertices and $E$ is the set of edges. Take a symmetric and doubly-stochastic *mixing matrix* $W = \{W_e\}_{e\in E}$, i.e. $0 \le W_{ij} \le 1$, $\sum_i W_{ij} = 1$ and $W^T = W$. The matrix $W$ encodes the graph topology since $W_{ij} = 0$ means no edge between the $i$th and the $j$th agents. Following [14], the optimization problem (1) can be rewritten as:

$$\underset{\mathbf{x}\in\mathbb{R}^{M\times P}}{\operatorname{minimize}}\, \frac{1}{M}\sum_{i=1}^{M} F_i(\mathbf{x}) \quad \text{subject to } W\mathbf{x} = \mathbf{x}\,, \tag{2}$$

where $\mathbf{x} = [x_1, ..., x_M]^\top$ and the functions $F_i$ are defined as $F_i(\mathbf{x}) := f_i(x_i)$.

### C. Overview of our approach

Our goal is to design a network of $N$ nodes and a mixing matrix $W$ such that state-of-the-art consensus optimization algorithms that apply multiplication of $W$ in the network, such as EXTRA [14], [15] and ADMM [16], [17], [18], can solve problem (1) with a nearly minimal amount of total communication. Our target accuracy is $\|z^k - z^*\| < \varepsilon$, where $k$ is the iteration index, $\varepsilon$ is a threshold, and $(z^k)_k$ is the sequence of iterates, and $z^*$ is its limit. We use $z^k, z^*$ instead of $x^k, x^*$ since EXTRA and ADMM iterate both primal and dual variables. **Let $U$ be the amount of *communication per iteration*, and $K$ be the number of iterations needed to reach the specified accuracy.** The *total communication* is their product $(UK)$. Apparently, $U$ and $K$ both depend on the network. A dense network usually implies a large $U$ and a small $K$. On the contrary, a highly sparse network typically leads to a small $U$ and a large $K$. (There are special cases such as the *star* network.) The optimal density is typically neither of them. Hence we shall find the optimal balance. To achieve

This work was supported in part by NSF grant ECCS-1462398. Emails: {ytchow,wuty11,wotaoyin}@math.ucla.edu and wilburs@bu.edu

this, we express $U$ and $K$ in the network parameters that we can tune, along with other parameters of the problem and the algorithm that affect total communication but we do not tune. Then, by finding the optimal values of the tuning parameters, we obtain the optimal network.

The dependence of $U$ and $K$ on the network can be complicated. Therefore, we make a series of simplifications to the problem and the network so that it becomes sufficiently simple to find the minimizer to $(UK)$. We mainly (but are not bound to) consider strongly convex objective functions and decentralized algorithms using fixed step sizes (e.g., EXTRA and ADMM) because they have the linear convergence rate: $\|z^k - z^*\| \leq C^k$ for $C < 1$. Given a target accuracy $\varepsilon$, from $C^K = \varepsilon$, we deduce the sufficient number of iterations $K = \log(\varepsilon^{-1})/\log(C^{-1})$. Since the constant $C < 1$ depends on network parameters, so does $K$.

We mainly work with three network parameters: the condition number of the graph Laplacian, the first non-trivial eigenvalue of the incidence matrix, and the degree of each node. In this work, we restrict ourselves to the $d$-regular graph, e.g., every node is connected to exactly $d$ other nodes.

We further (but are not bound to) simplify the problem by assuming that all edges have the same cost of communication. Since most algorithms perform a fixed number (typically, one or two) of rounds of communication in each iteration along each edge, $U$ equals a constant times $d$.

So far, we have simplified our problem to minimizing $(Kd)$, which reduces to choosing $d$, deciding the topology of the $d$-regular graph, as well setting the mixing matrix $W$.

For the two algorithms EXTRA and ADMM, we deduce (in Section II from their existing analysis) that $K$ is determined by the mixing matrix $W$ and $\tilde{\kappa}(L_G)$, where $L_G$ denotes the graph Laplacian and $\tilde{\kappa}(L_G)$ is its reduced condition number. We simplify the formula of $K$ by relating $W$ to $L_G$ (thus eliminating $W$ from the computation). Hence, our problem becomes minimizing $(Kd)$, where $K$ only depends on $\tilde{\kappa}(L_G)$ and $\tilde{\kappa}(L_G)$ further depends on the degree $d$ (same for every node) and the topology of the $d$-regular graph.

By classical graph theories, a $d$-regular graph satisfies $\tilde{\kappa}(L_G) \leq \frac{d+\lambda_1(A)}{d-\lambda_1(A)}$, where $\lambda_1(A)$ is the first non-trivial eigenvalue of the incidence matrix $A$, which we define later. We minimize $\lambda_1(A)$ to reduce $\frac{d+\lambda_1(A)}{d-\lambda_1(A)}$ and thus $\tilde{\kappa}(L_G)$. Interestingly, $\lambda_1(A)$ is lower bounded by the Alon Boppana formula of $d$ (thus also depends on $d$). The lower bound is nearly attained by Ramanujan graphs, a type of graph with a small number of edges but having high connectivity and good spectral properties! Therefore, given a degree $d$, we shall choose Ramanujan graphs, because they approximately minimize $\lambda_1(A)$, thus $\tilde{\kappa}(L_G)$, and in turn $(Kd)$ and $(KU)$.

Unfortunately, Ramanujan graphs are not known for all values of $(N, d)$. For large $N$, we follow [19] to construct a random network, which has nearly minimal $\lambda_1(A)$. For small $N$, we apply the Ramanujan Sparsifier [20]. These approaches lead to near-optimal network topologies for a given $d$.

After the above deductions, minimizing the total communication simplifies to choosing the degree $d$. To do this, one must know the cost ratio between computation and communication, which depends on applications. Once these parameters are known, a simple one-dimensional search method will find $d$.

### D. Communication reduction

We find that, in different problems, the graph topology affects total communication to different extents. When one function $f_i(x)$ has a big influence to the consensus solution, total communication is sensitive to the graph topology. On the other hand, computing the consensus average of a set of similar numbers is insensitive to the graph topology, so a Ramanujan graph does not make much improvement. Therefore, our proposed graphs work only for the former type of problem and data. See section IV for more details and comparison.

### E. Limitations and possible resolutions

We assume nodes having the same degree, edges having the same communication cost, and a special mixing matrix $W$. Despite these assumptions, the Ramanujan graph may still serve as the starting point to reduce the total communication in more general settings, which is our future work.

### F. Related work

Other work that has considered and constructed Ramanujan graphs include [21], [22]. Graph design for optimal convergence rate under decentralized consensus averaging and optimzation is considered in [3], [23], [24], [25], [26], [27], [28]. However, we suspect the mixing matrices $W$ recommended in the above for average consensus might sometimes not be the best choices for decentralized optimization, since consensus averaging is a lot less communication demanding than decentralized optimization problems (c.f. Section I-D), and it is the least sensitive to adding, removing, or changing one node. On the other hand, we agree for consensus averaging, those matrices recommended serve as the best candidates.

## II. Communication cost in decentralized optimization

As described in Subsection I-C, we focus on a special class of decentralized algorithm that allows us to optimize its communication cost by designing a suitable graph topology.

For the notational sake, let us recall several definitions and properties of a graph. See [29], [30], [31] for background. Given a graph $G$, the adjacency matrix $A$ of $G$ is a $|V| \times |V|$ matrix with components

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E\,, \\ 0 & \text{if otherwise.} \end{cases}$$

An equally important (oriented) incidence matrix is defined as is a $|E| \times |V|$ matrix such that

$$B_{ei} = \begin{cases} 1 & \text{if } e = (i,j) \text{ for some } j, \text{ where } j > i\,, \\ -1 & \text{if } e = (i,j) \text{ for some } j, \text{ where } j < i\,, \\ 0 & \text{otherwise.} \end{cases}$$

The degree of a node $i \in E$, denoted as $\deg(i)$, is the number of edges incident to the node. Every node in a $d$-regular graph has the same degree $d$. The graph Laplacian of $G$ is:

$$L_G := D - A = B^T B\,,$$

where $D$ has diagonal entries $D_{ii} = \deg(i)$ and 0 elsewhere. The matrix $L_G$ is symmetric and positive semidefinite. The multiplicity of its trivial eigenvalue 0 is the number of connected components of $G$. Therefore, we consider its first *nonzero* eigenvalue, denoted as $\widetilde{\lambda_{\min}}(L_G)$ and its largest eigenvalue, denoted as $\lambda_{\max}(L_G)$. The reduced condition number of $L_G$ is defined as:

$$\tilde{\kappa}(L_G) := \lambda_{\max}(L_G)/\widetilde{\lambda_{\min}}(L_G)\,, \tag{3}$$

which is of crucial importance to our exposition. We need a weighted graph and its graph Laplacian. A weighted graph $G = (V, E, w)$ is a graph with nonnegative edge weights $w = (w_e)$. Its graph Laplacian is defined as

$$L_{G,w} := B^T \mathrm{Diag}(w) B\,.$$

Its eigenvalues and condition numbers are similarly defined, e.g., $\tilde{\kappa}(L_{G,w})$ being its reduced condition number.

Next we define a special class of decentralized algorithm, which simplifies the computation of total communication cost.

*Definition 1:* We call a first-order decentralized algorithm to solve (2) communication-optimizable if any sequence $\{\mathbf{z}^k\}$ generated by the algorithm satisfies

$$\|\mathbf{z}^k - \mathbf{z}^*\| \le R(\Theta, \Gamma_f, \tilde{\kappa}(L_{G,w}), k)\,, \tag{4}$$

for a certain norm $\|\cdot\|$ and function $R$, where $\Theta$ consists of all algorithmic parameters specified by the user, $\Gamma_f$ is a constant depending on the objective function $f$. In addition, $R$ is non-increasing as $\tilde{\kappa}(L_{G,w})$ decreases or $k$ increases.

The above definition means that there exists a convergence-speed bound that depends on only $L_{G,w}$ and $k$ while other parameters $\Theta, \mu_f$ are fixed. This description of convergence rate is met by many algorithms mainly because $\tilde{\kappa}(L_{G,w})$ reflects the connectivity of graph topology. In fact, if $G$ is $d$-regular and weighted, then we have from [30], [31]:

$$\tilde{\kappa}(L_G) \le \frac{d + \lambda_1(A)}{d - \lambda_1(A)},$$

where $\lambda_1(A) \ne d$ is the first non-trivial eigenvalue of the adjacency matrix $A$ (as the eigenvalue $\lambda_0(A) = d$ is trivial and corresponds to the eigenvalue 0 of $L_G$) and is related to the connectivity of $G$ by the Cheeger inequalities [32]:

$$\tfrac{1}{2}(d - \lambda_1(A)) \le h(G) \le \sqrt{2d(d - \lambda_1(A))}$$

where $h(G)$ is the Cheeger constant quantifying the connectivity of $G$. Therefore saying that $R(\Theta, \mu_f, \tilde{\kappa}(L_{G,w}), k)$ is nondecreasing with $\tilde{\kappa}(L_{G,w})$ is related to saying that is nonincreasing with the connectivity of $G$. (Higher connectivity lets the algorithm converge faster.) Hence, we hope to increase the number of edges $E$ of $G$ though a cost is associated.

With the help of $L_{G,w}$, for simplicity we might choose the mixing matrix $W$ to take the form [23], [24]:

$$W_{L_{G,w}} := I - \frac{2}{(1+\Theta_1)\lambda_{\max}(L)} L_{G,w}\,, \text{ for } 0 < \Theta_1 < 1\,, \tag{5}$$

where $\Theta_1$ is among the set of algorithmic parameters $\Theta$ mentioned above. This choice of $W$ is symmetric and doubly stochastic. Other choices are given in [25], [26], [27], [28], [3]. Under our choice of $W$, there are many examples of communication-optimizable algorithms, where the convergence rate depends only on $\tilde{\kappa}(L_{G,w})$. The convergence rate of EXTRA [14], [15] can be explicitly represented, after some straight-forward simplifications by letting $W = W_{L_{G,w}}$ (5):

$$\frac{\|\mathbf{z}^k - \mathbf{z}^*\|}{\|\mathbf{z}^0 - \mathbf{z}^*\|} \le \Big(1 + \min\big\{\tfrac{1}{p\,\tilde{\kappa}(L_{G,w}) + q}, \tfrac{1}{r\tilde{\kappa}(L_{G,w})}\big\}\Big)^{-\frac{k}{2}}\,,$$

for some norm $\|\cdot\|$ (where $\mathbf{z}$ is related to $\mathbf{x}$), and decentralized ADMM [16], [17], [18]:

$$\frac{\|\mathbf{z}^k - \mathbf{z}^*\|}{\|\mathbf{z}^0 - \mathbf{z}^*\|} \le \Big(1 + \sqrt{p\,\tilde{\kappa}(L_{G,w})^{-2} + q} - \sqrt{q}\Big)^{-\frac{k}{2}}\,,$$

where $p, q, r$ are some coefficients depending only on $\Theta, \mu_f$. Apparently, while fixing other parameters, a smaller $\tilde{\kappa}(L_{G,w})$ tends to give faster convergence. We see the same in extension of EXTRA, PG-EXTRA [15], for nonsmooth functions.

In order to consider the communication cost for an optimization procedure, we need the following two quantities. The first one is the amount of communication per iteration $U$:

$$U(\mu, W) := \textstyle\sum_{e\in E} \mu_e |W_e|_0, \tag{6}$$

where $\mu = (\mu_e)$ the communication cost on each edge $e$ and $|\cdot|_0$ as the 1/0 function that indicates if $e$ is used for communication or not. When $W$ is related to $L_{G,w}$ via (5), we also write $U(\mu, L_{G,w}) := U(\mu, W_{L_{G,w}})$ to represent this quantity. Next, we introduce the second quantity, the total communication cost for an optimization procedure. For this purpose, let us recall that $\Gamma_f$ represents a set of constants depending only on the structure of the target function $f$, and $\Theta$ be a set of parameters not depending on either $f$ of the graph $G$. We let $K$ be the number of iterations to achieve the accuracy $\|\mathbf{z}^k - \mathbf{z}^*\| \le \epsilon$,

$$K = K(\Theta, \Gamma_f, \tilde{\kappa}(L_{G,w}), \epsilon),$$

as $K$ clearly depends on $\Gamma_f$ , $\Theta$, $\tilde{\kappa}(L_{G,w})$, and $\epsilon$. By the definition of $R$ in (4), we have

$$R\big(\Theta, \Gamma_f, \tilde{\kappa}(L_{G,w}), K(\Theta, \Gamma_f, \tilde{\kappa}(L_{G,w}), \epsilon)\big) = \epsilon\,.$$

*Theorem 1:* The total communication cost for $(\mathbf{z}^i)_{i=0}^k$ generated by a communication-optimizable algorithm to satisfy $\|\mathbf{z}^k - \mathbf{z}^*\| \le \varepsilon$, denoted as $J(\Theta, \Gamma_f, W_{L_{G,w}}, \epsilon)$, satisfies:

$$J(\Theta, \Gamma_f, W_{L_{G,w}}, \epsilon) \le K\big(\Theta, \Gamma_f, \tilde{\kappa}(L_{G,w}), \epsilon\big) \cdot U\big(\mu, L_{G,w}\big)\,. \tag{7}$$

The cost consists of two parts, the first part $K$ decreasing with the connectivity of $G$ while the other part $U$ growing with it.

In general, how to optimize $J$ depends on each algorithm (which gives the function $K$) and the weights $\{\mu_e\}$. Given an additional assumption that $\{\mu_e\}$ are uniformly, we can assume

$$U(\mu, L_{G,w}) \approx F(d^{\text{ave}})\,, \tag{8}$$

where $d^{\text{ave}} = \sum_i \deg(i)/\sum_i 1$ is the average degree, and $F$ a function that increases monotonically w.r.t. $d^{\text{ave}}$. Therefore, a reasonable approximation of the communication efficiency optimization problem becomes

$$\min_G J(\Theta, \Gamma_f, W_{L_{G,w}}, \epsilon) \approx \min_{d^{\text{ave}}} \{H(d^{\text{ave}})F(d^{\text{ave}})\} \tag{9}$$

where $H(d^{\text{ave}})$ reads

$$H(d^{\text{ave}}) := \min_{(G_{d^{\text{ave}}},w)} K\big(\Theta, \Gamma_f, \tilde{\kappa}(L_{G_{d^{\text{ave}}},w}), \epsilon\big), \tag{10}$$

where $G_{d^{\text{ave}}}$ denotes an unknown graph whose nodes have the same degree. Optimizing $d^{\text{ave}}$ in (9) after knowing $H(d^{\text{ave}})$ and $F(d^{\text{ave}})$ can only be done on a problem-by-problem and algorithm-by-algorithm basis. However, the minimum arguments in the expression (10) in the definition of $H(d^{\text{ave}})$ are always graphs such that $\tilde{\kappa}(L_{G_{d^{\text{ave}}},w})$ is minimized. In light of this, we propose to promote communication efficiency by optimizing $\tilde{\kappa}(L_{G_{d^{\text{ave}}},w})$ in any case (i.e. whether we know the actual expression of $K$ or not, or even in the case when $\{\mu_e\}$ is not so uniform.)

## III. Graph optimization (of $\tilde{\kappa}(L_G)$ or $\tilde{\kappa}(L_{G,w})$)

### *A. Exact optimizer with specific nodes and degree $(N, d)$: Ramanujan graphs*

For a general $d$-regular graph, it is known from Alon Boppana Theorem in [33], [34], [35] that

$$\lambda_1(A) \geq 2\sqrt{d-1} - \frac{2\sqrt{d-1}-1}{\lfloor D/2 \rfloor}$$

where $D$ is the diameter of $G$.

*Definition 2:* [29], [30], [31], [36] A $d$-regular graph $G$ is a Ramanujan graphs if it satisfies

$$\lambda_1(A_G) \leq 2\sqrt{d-1}, \tag{11}$$

where $A_G$ is the adjacency matrix of $G$.

In fact, a Ramanujan graph serves as an asymptotic minimizer of $\tilde{\kappa}(L_{G_d})$ for a $d$-regular graph $G_d$. For a Ramanujan graph, the reduced condition number of the Laplacian satisfies:

$$\tilde{\kappa}(L_{G_d}) \leq \frac{d+2\sqrt{d-1}}{d-2\sqrt{d-1}}\,.$$

If we use a Ramanujan graph that minimizes $\tilde{\kappa}(L_{G_d})$, we can then find an approximate minimizer in (10) and thus (9) for the total communication cost by further finding $d$ (or $d^{\text{ave}}$).

Explicit constructions of Ramanujan graphs are known only for some special $(N, d)$, as labelled Cayley graphs and the Lubotzky-Phillips-Sarnak construction [37], with some computation given in [38], [39] or construction of bipartite Ramanujan graphs [40]. Interested readers may consult [30], [29], [41], [31], [37], [36] for more references.

### *B. Optimizer for large $N$: Random $d$-regular graphs*

In some practical situations, however, we encounter pairs $(N, d)$ where an explicit Ramanujan graph is still unknown. When $N$ is very large, we propose a random $d$-regular graph as a solution of optimal communication efficiency, with randomness performed as follows [19] (where $d$ shall be even and $N$ can be an arbitrary integer that is greater than 2): choosing independently $d/2$ permutations $\pi_j, j = 1, .., d/2$ of the numbers from 1 to $n$, with each permutation equally likely, a graph $G = (V, E)$ with vertices $V = 1, ..., n$ is given by

$$E = \{(i, \pi_j(i)), i = 1, ...n, j = 1, ..., d/2\}\,.$$

The validity of using random graphs as an approximate optimizer is ensured by the Friedman Theorem as below:

*Theorem 2:* [19] For every $\epsilon > 0$,

$$\mathbb{P}(\lambda_1(A) \leq 2\sqrt{d-1} + \epsilon) = 1 - o_N(1) \tag{12}$$

where $G$ is a random $(N, d)$-graph.[1]
Roughly speaking, this theorem says that for a very large $N$, almost all random $(N, d)$ graphs are Ramanujan, i.e. they are nicely connected. Therefore, it is just fine to adopt a random $(N, d)$ graph when $N$ is too large for a Ramanujan graph or sparsifier to be explicitly computed.

It is possible that some regular graphs are not connected. To address this issue, we first grow a random spanning tree of the underlying graph to ensure connectivity. A random generator for regular graphs is GGen2 [42].

### *C. Approximation for small $N$: 2-Ramanujan Sparsifier*

For some small $N$, an explicit $(N, d)$-Ramanujan graph may still be unavailable. We hope to construct an approximate Ramanujan graph, i.e. expanders which are sub-optimal but work well in practice. An example is given by the 2-Ramanujan sparsifier, which is the subgraph $H$ as follows:

*Theorem 3:* [20] For every $d > 1$, every undirected weighted graph $G = (V, E, w)$ on $N$ vertices contains a weighted subgraph $H = (V, F, w')$ with at most $d(N-1)$ edges (i.e. an average at most $2d$ graph) that satisfies

$$L_{G,w} \preccurlyeq L_{H,w'} \preccurlyeq \frac{d+1+2\sqrt{d}}{d+1-2\sqrt{d}}\, L_{G,w} \tag{13}$$

where the Loewner partial order relation $A \preccurlyeq B$ indicates that $B - A$ is a positive semidefinite matrix.
This theorem allows us to construct a sparsifier from an original graph $G$. Actually, the proof of the theorem provides us with an explicit algorithm for such a construction [20].

## IV. Numerical Experiments

In this subsection, we illustrate the communication efficiency optimization achieved by expander graphs.

Since the focus of our work is not to investigate various methods to produce expander graphs, we did not test **Algorithm I**. Rather, our focus is to illustrate that a smaller *reduced condition number* improves communication efficiency and that expander graphs are good choices to reduce communication in decentralized optimization. Therefore we compare the difference in communication efficiency using existing graphs with different reduced condition numbers. We compare the convergence speeds on a family of (possibly non-convex non-smooth) problems which are approximate formulations of finding the sup-norm of a vector $l = (l_1, ..., l_M)$ (up to sign).
**Example 1**. We choose $M = P = 1092$. Our problem is:

$$\min_{x \in \mathbb{R}^P} f(x) = \tfrac{1}{M} \textstyle\sum_{i=1}^{M} f_i(x),$$

where

$$f_i(x) = l_i x_i / (\|x\|_1 + \varepsilon)$$

[1] The notion $g(N) = o_N(f(N))$ means $\lim_{N\to\infty} g(N)/f(N) = 0$.

| | $\tilde{\kappa}(L_G)$ | No. of Itr. | Total comm. complexity |
|---|---|---|---|
| LPS(29, 13) | 1.9538 | 52 | 1703520 |
| Regular 30 graph | 30.5375 | 444 | 14545440 |
| Regular 60 graph | 32.1103 | 494 | 32366880 |
| Regular 120 graph | 27.1499 | 454 | 59492160 |

TABLE I: Communication complexity in **Example 1** for Ramanujan 30-graph $\mathrm{LPS}(29, 13)$, 30-reg, 60-reg, and120-reg graph.

for a small $\varepsilon$ chosen as $\varepsilon = 1 \times 10^{-5}$. One very important remark is that this optimization problem is non-convex and non-smooth, and the $\varepsilon$-optimizer occupies a tabular neighbourhood of a whole ray $\{-\lambda e_n : \lambda > 0\}$, where $l_n = \|l\|_\infty$ and $e_n$ is the $n$th coordinate vector $(0, 0, ..., 1, ...., 0)$.

We solve the problem using the EXTRA algorithm [14]:

$$\begin{cases} \mathbf{x}^1 = W\mathbf{x}^0 - \alpha\nabla\mathbf{f}(\mathbf{x}^0)\,, \\ \mathbf{x}^{k+2} = (W+I)\mathbf{x}^{k+1} - \frac{W+I}{2}\mathbf{x}^k - \alpha[\nabla\mathbf{f}(\mathbf{x}^{k+1}) - \nabla\mathbf{f}(\mathbf{x}^k)]\,. \end{cases}$$

To compare speeds, we plot the following sequence:

$$\delta_k := \tfrac{1}{M}\textstyle\sum_{i=1}^M F_i(\mathbf{x}^k) - \min_x F(x)\,.$$

Figure 1 shows the sequence $\delta_k$ under 4 different graph topologies: Ramanujan 30-graph $\mathrm{LPS}(29, 13)$, a regular-30, a regular-60 graph, and a regular-120 graph. Our regular $d$ graphs (other than $LPS(29, 13)$) are generated by joining the $i$th node to the $\{i + \lfloor\frac{N}{d}\rfloor k \mod N : 0 \le k < \frac{d}{2}\}$th nodes for all $0 \le i \le N - 1$ (in here we label the nodes from 0 to $N - 1$). The mixing matrices $W$ that we use are generated as $W_{L_{G,w}}$ as described in (5) with $w = (w_e)$ where $w_e = 1$ on all edges $e \in E$. We clearly see that the convergence rate of the algorithm under Ramanujan 30-graph $\mathrm{LPS}(29, 13)$ is even better than that of a regular-120 graph in the first 40 iterations. Afterward, the curve of the Ramanujan 30-graph becomes flat, probably because it already arrived a small tabular neighborhood of the ray of $\varepsilon$-optimizer. Table I shows the number of iteration $k_0$ as well as its total communication complexity to achieve

$$\text{stopping rule:} \quad |\delta_{k_0} - \delta_{k_0-1}| < 1 \times 10^{-3}.$$

We see that an expander graph reduces communication complexity. One can observe the rapid drop of the red curve because of its efficiency to locate the active set.

**Example 2**. In this case we again choose $M = 1092$, but $P = 1$ (i.e. 1 dimensional problem). Our problem is with a similar form as in **Example 1**, but with

$$f_i(x) = \chi_{a \ge \sqrt{|l_i|}}(x) + |x|^2.$$

This is a convex non-smooth problem. We apply PG-EXTRA [15] to solve this problem. To proceed, we split $f_i$ as

$$\begin{cases} s_i(x) := |x|^2 \\ r_i(x) := \chi_{a \ge \sqrt{|l_i|}}(x), \end{cases}$$

and apply the PG-EXTRA iteration:

$$\begin{cases} \mathbf{x}^{\frac{1}{2}} = W\mathbf{x}^0 - \alpha\nabla\mathbf{f}(\mathbf{x}^{k+1})\,, \\ \mathbf{x}^1 = \arg\min \mathbf{r}(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}^{\frac{1}{2}}\|_2^2\,, \\ \mathbf{x}^{k+\frac{3}{2}} = W\mathbf{x}^{k+1} + \mathbf{x}^{k+\frac{1}{2}} - \frac{W+I}{2}\mathbf{x}^k - \alpha[\nabla\mathbf{s}(\mathbf{x}^{k+1}) - \nabla\mathbf{s}(\mathbf{x}^k)]\,, \\ \mathbf{x}^{k+2} = \arg\min \mathbf{r}(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}^{k+\frac{3}{2}}\|_2^2\,. \end{cases}$$

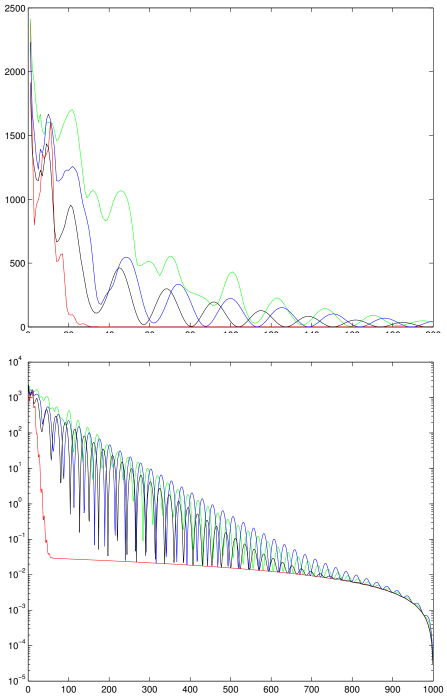

Fig. 1: Convergence rate of EXTRA in **Example 1** over Ramanujan 30-graph $\mathrm{LPS}(29, 13)$ (red, the best), 30-reg graph (blue), 60-reg graph (green) and 120-reg graph (black).

| | $\tilde{\kappa}(L_G)$ | No. of Itr. | Total comm. complexity |
|---|---|---|---|
| LPS(29, 13) | 1.9538 | 811 | 26568360 |
| Regular 30 graph | 30.5375 | 1001 | 32792760 |
| Regular 60 graph | 32.1103 | 832 | 54512640 |
| Regular 120 graph | 27.1499 | 953 | 124881120 |

TABLE II: Communication complexity in **Example 2** for Ramanujan 30-graph $\mathrm{LPS}(29, 13)$, 30-reg, 60-reg and120-reg graphs.

To compare speeds, we plot the following sequence:

$$\delta_k := \left|\tfrac{1}{M}\textstyle\sum_{i=1}^M s_i(\mathbf{x}_i^k) - \min_x F(x)\right|.$$

and exclude the $r_i(x)$ part. The mixing matrices that we use are the same as those described in **Example 1**. As shown in Figure 2 the convergence rate of the algorithm under the Ramanujan 30-graph $\mathrm{LPS}(29, 13)$ is still the best though is less outstanding. Table II shows the numbers of iteration $k_0$ as well as the total communication complexities to achieve $|\delta_{k_0} - \delta_{k_0-1}| < 1 \times 10^{-3}$. The expander graph has the clear advantage.

## References

[1] Alexandros G Dimakis, Soummya Kar, Jose MF Moura, Michael G Rabbat, and Anna Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[2] Bjorn Johansson, "On distributed optimization in networked systems," 2008.

[3] Lin Xiao, Stephen Boyd, and Seung-Jean Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[4] Pedro A Forero, Alfonso Cano, and Georgios B Giannakis, "Consensus-based distributed support vector machines," *The Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, 2010.

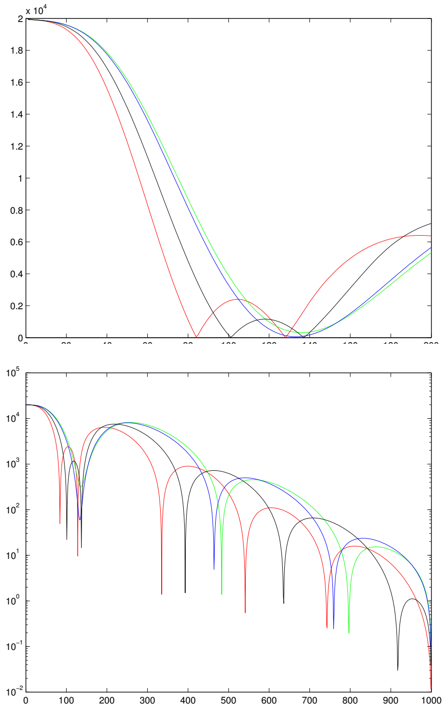

Fig. 2: Convergence rate of PG-EXTRA in **Example 2** over Ramanujan 30-graph LPS$(29, 13)$ (in red), 30-reg (in blue), 60-reg (in green), and 120-reg graph (in black).

[5] Gonzalo Mateos, Juan Andres Bazerque, and Georgios B Giannakis, "Distributed sparse linear regression," *Signal Processing, IEEE Transactions on*, vol. 58, no. 10, pp. 5262–5276, 2010.

[6] Joel B Predd, Sanjeev R Kulkarni, and H Vincent Poor, "A collaborative training algorithm for distributed learning," *Information Theory, IEEE Transactions on*, vol. 55, no. 4, pp. 1856–1871, 2009.

[7] Juan Andres Bazerque and Georgios B Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *Signal Processing, IEEE Transactions on*, vol. 58, no. 3, pp. 1847–1862, 2010.

[8] Juan Andres Bazerque, Gonzalo Mateos, and Georgios B Giannakis, "Group-lasso on splines for spectrum cartography," *Signal Processing, IEEE Transactions on*, vol. 59, no. 10, pp. 4648–4663, 2011.

[9] Vassilis Kekatos and Georgios Giannakis, "Distributed robust power system state estimation," *Power Systems, IEEE Transactions on*, vol. 28, no. 2, pp. 1617–1626, 2013.

[10] Qing Ling and Zhi Tian, "Decentralized sparse signal recovery for compressive sleeping wireless sensor networks," *Signal Processing, IEEE Transactions on*, vol. 58, no. 7, pp. 3816–3827, 2010.

[11] Qing Ling, Zaiwen Wen, and Wotao Yin, "Decentralized jointly sparse optimization by reweighted minimization," *Signal Processing, IEEE Transactions on*, vol. 61, no. 5, pp. 1165–1170, 2013.

[12] Kun Yuan, Qing Ling, Wotao Yin, and Alejandro Ribeiro, "A linearized bregman algorithm for decentralized basis pursuit," in *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*. IEEE, 2013, pp. 1–5.

[13] Qing Ling, Yangyang Xu, Wotao Yin, and Zaiwen Wen, "Decentralized low-rank matrix completion," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2925–2928.

[14] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[15] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin, "A proximal gradient algorithm for decentralized nondifferentiable optimization," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 29642968, 2015.

[16] Euhanna Ghadimi, Andre Teixeira, Michael G Rabbat, and Mikael Johansson, "The admm algorithm for distributed averaging: Convergence rates and optimal parameter selection," in *2014 48th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2014, pp. 783–787.

[17] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[18] Joao FC Mota, Joao MF Xavier, Pedro MQ Aguiar, and Markus Puschel, "D-admm: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[19] Joel Friedman, *A proof of Alon's second eigenvalue conjecture and related problems*, American Mathematical Soc., 2008.

[20] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava, "Twice-ramanujan sparsifiers," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1704–1721, 2012.

[21] Yoonsoo Kim and Mehran Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph laplacian," *IEEE transactions on Automatic Control*, vol. 51, no. 1, pp. 116–120, 2006.

[22] Arpita Ghosh and Stephen Boyd, "Growing well-connected graphs," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 6605–6611.

[23] Lin Xiao and Stephen Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[24] Xiaochuan Zhao, Sheng-Yuan Tu, and Ali H Sayed, "Diffusion adaptation over networks under imperfect information exchange and non-stationary data," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3460–3475, 2012.

[25] Stephen Boyd, Persi Diaconis, and Lin Xiao, "Fastest mixing markov chain on a graph," *SIAM review*, vol. 46, no. 4, pp. 667–689, 2004.

[26] Kun Yuan, Qing Ling, and Wotao Yin, "On the convergence of decentralized gradient descent," *arXiv preprint arXiv:1310.7063*, 2013.

[27] John Nikolas Tsitsiklis, "Problems in decentralized decision making and computation.," Tech. Rep., DTIC Document, 1984.

[28] I-An Chen et al., *Fast distributed first-order methods*, Ph.D. thesis, Massachusetts Institute of Technology, 2012.

[29] M Ram Murty, "Ramanujan graphs," *Journal-Ramanujan Mathematical Society*, vol. 18, no. 1, pp. 33–52, 2003.

[30] Michael William Newman, *The Laplacian spectrum of graphs*, Ph.D. thesis, University of Manitoba, 2000.

[31] Fan RK Chung, *Spectral graph theory*, vol. 92, American Mathematical Soc., 1997.

[32] Noga Alon and Joel H Spencer, *The probabilistic method*, John Wiley & Sons, 2004.

[33] Noga Alon, "On the edge-expansion of graphs," *Combinatorics, Probability and Computing*, vol. 6, no. 02, pp. 145–152, 1997.

[34] Noga Alon, Alexander Lubotzky, and Avi Wigderson, "Semi-direct product in groups and zig-zag product in graphs: connections and applications," in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE, 2001, pp. 630–637.

[35] Jozef Dodziuk, "Difference equations, isoperimetric inequality and transience of certain random walks," *Transactions of the American Mathematical Society*, vol. 284, no. 2, pp. 787–794, 1984.

[36] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak, "Ramanujan graphs," *Combinatorica*, vol. 8, no. 3, pp. 261–277, 1988.

[37] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak, "Explicit expanders and the ramanujan conjectures," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. ACM, 1986, pp. 240–246.

[38] "Lubotzky-phillips-sarnak graphs," http://www.mast.queensu.ca/~ctardif/LPS.html, Accessed: 2010-09-30.

[39] Randy Elzinga, "Producing the graphs of lubotzky, phillips and sarnak in matlab," http://www.mast.queensu.ca/~ctardif/lps/LPSSup.pdf, 2010.

[40] Adam Marcus, Daniel A Spielman, and Nikhil Srivastava, "Interlacing families i: Bipartite ramanujan graphs of all degrees," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 529–537.

[41] Shlomo Hoory, Nathan Linial, and Avi Wigderson, "Expander graphs and their applications," *Bulletin of the American Mathematical Society*, vol. 43, no. 4, pp. 439–561, 2006.

[42] Wei Shi, "Ggen: Graph generation for network computing simulations," http://home.ustc.edu.cn/~shiwei00/html/GGen.html, 2014.