Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization
Angelia Nedić, Alex Olshevsky, and Michael G. Rabbat

Abstract
In decentralized optimization, nodes cooperate to minimize an overall objective function that is the sum (or average) of per-node private objective functions. Algorithms interleave local computations with communication among all or a subset of the nodes. Motivated by a variety of applications—distributed estimation in sensor networks, fitting models to massive data sets, and distributed control of multi-robot systems, to name a few—significant advances have been made towards the development of robust, practical algorithms with theoretical performance guarantees. This paper presents an overview of recent work in this area. In general, rates of convergence depend not only on the number of nodes involved and the desired level of accuracy, but also on the structure and nature of the network over which nodes communicate (e.g., whether links are directed or undirected, static or time-varying). We survey the state-of-the-art algorithms and their analyses tailored to these different scenarios, highlighting the role of the network topology.

I. INTRODUCTION
In multi-agent consensus optimization, n agents or nodes, as we will refer to them throughout this article, cooperate to solve an optimization problem. A local objective function \( f_i : \mathbb{R}^d \rightarrow \mathbb{R} \) is associated with each node \( i = 1, \ldots, n \), and the goal is for all nodes to find and agree on a minimizer of the average objective \( f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \) in a decentralized way. Each node maintains its own copy \( x_i \in \mathbb{R}^d \) of the optimization variable, and node \( i \) only has direct access to information about its local objective \( f_i \); for example, node \( i \) may be able to calculate the gradient \( \nabla f_i(x_i) \) of \( f_i \) evaluated at \( x_i \). Throughout this article we will focus on the case where the functions \( f_i \) are convex (so \( f \) is also convex) and where \( f \) has a non-empty set of minimizers so that the problem is well-defined.

Because each node only has access to local information, the nodes must communicate over a network to find a minimizer of \( f(x) \). Multi-agent consensus optimization algorithms are iterative, where each iteration typically involves some local computation followed by communication over the network. In most applications of interest, either it is not possible or one does not allow each node to communicate with every other node. The connectivity of the network (i.e., which pairs of nodes may communicate directly with each other) is represented as a graph with \( n \) vertices and with an edge between a pair of vertices if they communicate directly. In general, the network connectivity may change from iteration to iteration. We will see that, indeed, the communication network topology plays a key role in the convergence theory of multi-agent optimization methods in that it may limit the flow of information between distant nodes and thereby hinder convergence.

During the past decade, multi-agent consensus optimization has been the subject of intense interest, motivated by a variety of applications.

A. Nedić is with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, USA.
A. Olshevsky is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA.
M.G. Rabbat is with Facebook AI Research, Montréal, Canada, and the Department of Electrical and Computer Engineering, McGill University, Montréal, Canada.
Email: angelia.nedich@asu.edu, alexols@bu.edu, michael.rabbat@mcgill.ca
The work of A.N. and A.O. was supported by the Office of Naval Research under grant number N000014-16-1-2245. The work of A.O. was also supported by NSF under award CMMI-1463262 and AFOSR under award FA-95501510394. The work of M.R. was supported by the Natural Sciences and Engineering Research Council of Canada under awards RGPIN-2012-341596 and RGPIN-2017-06266.
A. Motivating Applications

The general multi-agent optimization problem described above was originally introduced and studied in the 1980’s in the context of parallel and distribute numerical methods [1]–[3]. The surge of interest in multi-agent convex optimization during the past decade has been fueled by a variety of applications where a network of autonomous agents must coordinate to achieve a common objective. We describe three such examples next.

1) Decentralized estimation: Consider a wireless sensor network with \( n \) nodes where node \( i \) has a measurement \( \zeta_i \) which is modeled as a random variable with density \( p(\zeta_i|x) \). In many applications of sensor networks, uncertainty is primarily due to thermal measurement noise introduced at the sensor itself, and so it is reasonable to assume that the observations \( \{\zeta_i\}_{i=1}^n \) are conditionally independent given the model parameters \( x \). In this case, the maximum likelihood estimate of \( x \) can obtained by solving

\[
\min_x \sum_{i=1}^n \log p(\zeta_i|x),
\]

which can be addressed using multi-agent consensus optimization methods [4]–[6] with \( f_i(x) = \log p(\zeta_i|x) \).

When the nodes communicate over a wireless network, whether or not a given pair of nodes can directly communicate is typically a function of their physical proximity as well as other factors (e.g., fading, shadowing) affecting the wireless channel, which may possibly result in time-varying and directed network connectivity.

2) Big data and machine learning: Many methods for supervised learning (e.g., classification or regression) can also be formulated as fitting a model to data. This task may generally be expressed as finding model parameters \( x \) by solving

\[
\min_x \sum_{j=1}^m l_j(x),
\]

where the loss function \( l_j(x) \) measures how well the model with parameters \( x \) describes the \( j \)th training instance, and the training data set contains \( m \) instances in total. For many popular machine learning models—including linear regression, logistic regression, ridge regression, the LASSO, support vector machines, and their variants—the corresponding loss function is convex [7].

When \( m \) is large, it may not be possible to store the training data on a single server, or it may be desirable for other reasons to partition the training data across multiple nodes (e.g., to speedup training by exploiting parallel computing resources, or because the data is being gathered and/or stored at geographically distant locations). In this case, the training task (1) can be solved using multi-agent optimization with local objectives of the form [8]–[12]

\[
f_i(x) = \sum_{j \in J_i} l_j(x),
\]

where \( J_i \) is the set of indices of training instances at node \( i \).

In this setting, where nodes typically correspond to servers communicating over a wired network, it may be feasible for every node to send and receive messages from all other nodes. However it is often still preferable, for a variety of reasons, to run multi-agent algorithms over a network with sparser connectivity. Communicating a message takes time, and reducing the number of edges in the communication graph at any iteration corresponds to reducing the number of messages to be transmitted. This results in iterations that take less time and also that consume less bandwidth.

Other architectures for distributed optimization have been studied, notably the master-worker architecture, where many workers perform calculations in parallel (e.g., of gradients or other step directions), and the master collects and applies these steps to the single copy of the optimization variable. In contrast to this architecture, which has a bottleneck and single point of failure at the master node, the multi-agent consensus approach can tolerate node or link failures—if one node fails, the remaining nodes in the network can continue to compute as before without any loss of information or the need to elect a new master node, since each has a local copy of the optimization variable.
3) Multi-robot systems: Similar to the previous example, multi-agent methods have attracted attention in applications requiring the coordination of multiple robots because they naturally lead to decentralized solutions. One well-studied problem arising in such systems is that of rendezvous—collectively deciding on a meeting time and location. When the robots have different battery levels or are otherwise heterogeneous, it may be desirable to design a rendezvous time and place, and corresponding control trajectories, which minimize the energy to be expended collectively by all robots. This can be formulated as a multi-agent optimization problem where the local objective \( f_i(x) \) at agent \( i \) quantifies the energy to be expended by agent \( i \) and \( x \) encodes the time and place for rendezvous [13], [14].

When robots communicate over a wireless network, the network connectivity will be dependent on the proximity of nodes as well as other factors affecting channel conditions, similar to in the first example. Moreover, as the robots move, the network connectivity is likely to change. It may be desirable to ensure that a certain minimal level of network connectivity is maintained while the multi-robot system performs its task, and such requirements can be enforced by introducing constraints in the optimization formulation [15].

B. Outline of the rest of the paper

The purpose of this article is to provide an overview of the main advances in this field, highlighting the state-of-the-art methods and their analyses, and pointing out open questions. During the past decade, a vast literature has amassed on multi-agent optimization and related methods, and we do not attempt to provide an exhaustive review (which, in any case, would not be feasible in the space of one article). Rather, in addition to describing the main advances and results leading to the current state-of-the-art, we also seek to provide an accessible survey of theoretical techniques arising in the analysis of multi-agent optimization methods.

Distributed averaging algorithms—where each node initially holds a number, and the aim is to calculate the average at every node—form a fundamental building block of multi-agent optimization methods. We review distributed averaging algorithms and their convergence theory in Sec. [II]. Sec. [III] discusses multi-agent optimization methods and theory in the setting of undirected communication networks. Sec. [IV] then describes how these methods can be extended to run over networks with directed connectivity via the push-sum approach. In both Secs. [III] and [IV] we limit our attention to methods for unconstrained optimization problems running in a synchronous manner. Sec. [V] discusses a variety of additional extensions, including how to incorporate various sorts of constraints and asynchronous methods, and it describes connections to various other branches of distributed optimization. We conclude in Sec. [VI] mentioning open problems.

C. Notation

Before proceeding, we summarize some notation that is used throughout the rest of this paper.

A matrix is called stochastic if it is nonnegative and the sum of the elements in each row equals one. A matrix is called doubly stochastic if, additionally, the sum of the elements in each column equals one. To a stochastic matrix \( A \in \mathbb{R}^{n \times n} \), we will associate the directed graph \( G_A \) with vertex set \{1, \ldots, n\} and edge set \( E_A = \{(i, j) \mid a_{ji} > 0\}\). Note that this graph might contain self-loops. For convenience, we will abuse notation slightly by using the matrix \( A \) and the graph \( G_A \) interchangeably; for example, we will say that \( A \) is strongly connected. Similarly, we will say that the matrix \( A \) is undirected if \((i, j) \in E_A\) implies \((j, i) \in E_A\) (i.e., if \( A \) is symmetric). Finally, we will use \([A]_{\alpha}\) to denote a thresholded matrix obtained from \( A \) by setting every element smaller than \( \alpha \) to zero.

Given a sequence of stochastic matrices \( A^0, A^1, A^2, \ldots \), we will use \( A^{k:l} \) to denote the product of elements \( k \) to \( l \) inclusive, i.e.,

\[
A^{k:l} = A^k A^{k-1} \cdots A^l.
\]

We will say that a matrix sequence is \( B \)-strongly-connected if the graph with vertex set \{1, \ldots, n\} and edge set

\[
\bigcup_{k=1}^{(l+1)B-1} E_{A^k}
\]

is strongly connected for each \( l = 0, 1, 2, \ldots \). Intuitively, we partition the iterations \( k = 1, 2, \ldots \) into consecutive blocks of length \( B \), and the sequence is \( B \)-strongly-connected when the graph obtained by unioning the edges within each block is always strongly connected. When the graph sequence is undirected, we will simply say \( B \)-connected.
The out-neighbors of node $i$ at iteration $k$ refers to the set of nodes that can receive messages from it, 

$$N_{i}^{\text{out},k} = \{ j \mid a_{ji}^k > 0 \},$$

and similarly, the in-neighbors of $i$ at iteration $k$ are the nodes from which $i$ receives messages, 

$$N_{i}^{\text{in},k} = \{ j \mid a_{ij}^k > 0 \}.$$ 

When the graph is not time-varying, we will simply refer to the out-neighbors $N_{i}^{\text{out}}$ and in-neighbors $N_{i}^{\text{in}}$. When the graph is undirected, the sets of in-neighbors and out-neighbors are identical, so we will simply refer to the neighbors $N_{i}$ of node $i$, or $N_{i}^{k}$ in the time-varying setting. The out-degree of node $i$ at iteration $k$ is defined as the cardinality of $N_{i}^{\text{out},k}$ and is denoted by $d_{i}^{\text{out},k} = |N_{i}^{\text{out},k}|$. Similarly, $d_{i}^{\text{in},k}$, $d_{i}^{\text{out}}$, $d_{i}^{\text{in}}$, $d_{i}$ and $d_{i}$ denote the cardinalities, respectively, of $N_{i}^{\text{in},k}$, $N_{i}^{\text{out}}$, $N_{i}^{\text{in}}$, $N_{i}^{k}$, and $N_{i}$.

II. Consensus over Undirected and Directed Graphs

This section reviews methods for distributed averaging that will form a key building block in our subsequent discussion of methods for multi-agent optimization.

A. Preliminaries: Results for averaging

We begin by examining the linear consensus process defined as

$$x^{k+1} = A^k x^k, \quad k = 0, 1, \ldots$$

(2)

where the matrices $A^k \in \mathbb{R}^{n \times n}$ are stochastic, and the initial vector $x^0 \in \mathbb{R}^n$ is given. Various forms of Eq. (2) can be implemented in a decentralized multi-agent setting, and these form the backbone of many distributed algorithms.

For example, consider a collection of nodes interconnected in a directed graph and suppose node $i$ holds the $i$'th coordinate of the vector $x^k$. Consider the following update rule: at step $k$, node $i$ broadcasts the value $x_i^k$ to its out-neighbors, receives values $x_j^k$ from its in-neighbors, and sets $x_i^{k+1}$ to be the average of the messages it has received, so that

$$x_i^{k+1} = \frac{1}{d_i^{\text{in},k}} \sum_{j \in N_{i}^{\text{in},k}} x_j^k.$$

This is sometimes called the equal neighbor iteration, and by stacking up the variables $x_i^k$ into the vector $x^k$ it can be written in the form of Eq. (2) with an appropriate choice for the matrix $A^k$.

Over undirected graphs, one popular choice of update rule is to set

$$x_i^{k+1} = x_i^k + \epsilon \sum_{j \in N_{i}^{\text{out}}} x_j^k - x_i^k,$$

where $\epsilon > 0$ is sufficiently small. Unfortunately, finding an appropriate choice of $\epsilon$ to guarantee convergence of this iteration can be bothersome (especially when the graphs are time-varying), and it generally requires knowing an upper bound on the degrees of nodes in the network.

Another possibility (when the underlying graphs are undirected) is the so-called Metropolis update

$$x_i^{k+1} = x_i^k + \sum_{j \in N_{i}^{\text{out}}} \frac{1}{\max\{d_i^{\text{in}}, d_j^{\text{out}}\}} (x_j^k - x_i^k).$$

(3)

The Metropolis update requires node $i$ to broadcast the values $x_i^k$ and its degree $d_i^k$ to its neighbors. As we will see later, this update possesses some nice convergence properties. Observe that the Metropolis update of Eq. (3) can be written in the form of Eq. (2) where the matrices $A^k$ are doubly stochastic.

Under certain technical conditions, the iteration of Eq. (2) results in consensus, meaning that all of the $x_i^k$ (for $i = 1, \ldots, n$) approach the same value as $k \to \infty$. We describe one such condition next. The key properties needed to ensure asymptotic consensus are that the matrices $A^k$ should exhibit sufficient connectivity and aperiodicity (in the long term). In the following, we use the shorthand $G^k$ for $G_{A^k}$, the graph corresponding to the matrix $A^k$. The starting point of our analysis is the following assumption.
Assumption 1. The sequence of directed graphs $G^0, G^1, G^2, \ldots$ is $B$-strongly-connected. Moreover, each graph $G^k$ has a self-loop at every node.

As the next theorem shows, a variation on this assumption suffices to ensure that the update of Eq. \ref{eq:2} converges to consensus.

**Theorem 1.** Suppose the sequence of stochastic matrices $A^0, A^1, A^2, \ldots$ has the property that there exists an $\alpha > 0$ such that the sequence of graphs $G^0, G^1, G^2, \ldots$ satisfies Assumption \ref{assumption:strongly_connected}. Then $x(t)$ converges to a limit in $\text{span}\{1\}$ and the convergence is geometric. Moreover, if all the matrices $A^k$ are doubly stochastic then for all $i = 1, \ldots, n$,

$$
\lim_{k \to \infty} x^k = \frac{\sum_{i=1}^{n} x_i^0}{n}.
$$

We now turn to the proof of this theorem, which while being reasonably short, still builds on a sequence of preliminary lemmas and definitions which we present first. Given a sequence of directed graphs $G^0, G^1, G^2, \ldots$, we say that node $b$ is reachable from node $a$ in time period $k : l$ if there exists a sequence of directed edges $e^k, e^{k-1}, \ldots, e^{l+1}, e^l$, such that: (i) $e^j$ is present in $G^j$ for all $j = l, \ldots, k$, (ii) the origin of $e^j$ is $a$, (iii) the destination of $e^k$ is $b$. Note that this is the same as saying that $[W^{k:l}]_{ba} > 0$ if the matrices $W^k$ are nonnegative with $[W^k]_{ij} > 0$ if and only if $(j, i)$ belongs to $G^k$. We use $N^{k:i}(a)$ to denote the set of nodes reachable from node $a$ in time period $k : l$.

The first lemma discusses the implications of Assumption \ref{assumption:strongly_connected} for products of the matrices $A^k$.

**Lemma 2.** Suppose $A^0, A^1, A^2, \ldots$ is a sequence of nonnegative matrices with the property that there exists $\alpha > 0$ such that the sequence of graphs $G^0, G^1, G^2, \ldots$ satisfies Assumption \ref{assumption:strongly_connected}. Then for any integer $l$,

$$
A^{(l+n)B-1: lB}
$$

is a strictly positive matrix. In fact, every entry of $A^{(l+n)B-1: lB}$ is at least $\alpha^{nB}$.

**Proof.** Consider the set of nodes reachable from node $i$ in time period $k_{\text{start}}$ to $k_{\text{finish}}$ in the graph sequence $G^0, G^1, G^2, \ldots$. Because each of these graphs has a self-loop at every node by Assumption \ref{assumption:strongly_connected}, the reachable set never decreases, i.e.,

$$
N^{k_{\text{finish}}:k_{\text{start}}}(i) \subset N^{k_{\text{finish}}+1:k_{\text{start}}}(i) \text{ for all } i, k_{\text{start}}, k_{\text{finish}}.
$$

A further immediate consequence of Assumption \ref{assumption:strongly_connected} is that if $N^{mB-1: lB}(i) \neq \{1, \ldots, n\}$, then $N^{(m+1)B-1: lB}(i)$ is strictly larger than $N^{mB-1: lB}(i)$, because during times $(m + 1)B - 1 : mB$ there is an edge in some $[G^t]_a$ leading from the set of nodes already reachable from $i$ to those not already reachable from $i$. Putting together these two properties, we obtain that from $lB$ to $(l+n)B - 1$ every node is reachable, i.e.,

$$
N^{(l+n)B-1: lB}(i) = \{1, \ldots, n\}.
$$

But since every non-zero entry of $[A^k]_a$ is at least $\alpha$ by construction, this implies that $A^{(l+n)B-1: lB} \geq \alpha^{nB}$, and the lemma is proved.

**Lemma 2** tells us that, over sufficiently long horizons, products of the matrices $A^k$ have entries bounded away from zero. The next lemma discusses what multiplication by such a matrix does to the spread of a vector.

**Lemma 3.** Suppose $W$ is a stochastic matrix, every entry of which is at least $\beta > 0$. If $v = Wu$ then

$$
\max_{i=1,\ldots,n} v_i - \min_{i=1,\ldots,n} v_i \leq (1 - 2\beta) \left( \max_{i=1,\ldots,n} u_i - \min_{i=1,\ldots,n} u_i \right).
$$

**Proof.** Without loss of generality, let us assume that the largest entry of $u$ is $u_1$ and the smallest entry of $u$ is $u_n$. Then, for $l \in \{1, \ldots, n\}$,

$$
v_l \leq (1 - \beta)u_1 + \beta u_n
$$

$$
v_l \geq \beta u_1 + (1 - \beta)u_n,
$$

so that for any $a, b \in \{1, \ldots, n\}$, we have

$$
v_a - v_b \leq (1 - \beta)u_1 + \beta u_n - (\beta u_1 + (1 - \beta)u_n)
$$

$$
= (1 - 2\beta)u_1 - (1 - 2\beta)u_n.
$$
With these two lemmas in place, we are ready to prove Theorem 1.

Proof of Theorem 1. Since we have assumed that $G_{\{A^0\}, \{A^1\}, \{A^2\}, \ldots}$ satisfy Assumption 1 by Lemma 2, we have that

$$
\left[A^{lB: (l+n)B-1}\right]_{ij} \geq \alpha^n B
$$

for all $l = 0, 1, 2, \ldots,$ and $i, j = 1, \ldots, n$. Applying Lemma 3 gives that

$$
\max_{i=1, \ldots, n} x_i^{(l+n)B} - \min_{i=1, \ldots, n} x_i^{(l+n)B} \leq \left(1 - 2\alpha^n B\right) \left(\max_{i=1, \ldots, n} x_i^{lB} - \min_{i=1, \ldots, n} x_i^{lB}\right).
$$

Applying this recursively, we obtain that $|x^k_a - x^k_b| \to 0$ for all $a, b \in \{1, \ldots, n\}$.

To obtain further that every $x^k_i$ converges, it suffices to observe that $x^k_i$ lies in the convex hull of the vectors $x^t$ for $t \leq k$. Finally, since each $A^k_i$ is doubly stochastic,

$$
\sum_{j=1, \ldots, n} x^{k+1}_j = 1^T x^{k+1} = 1^T A^k x^k = 1^T x^k = \sum_{j=1, \ldots, n} x^k_j,
$$

where $1$ denotes a vector with all entries equal to one, and thus all $x^k_i$ must converge to the initial average. □

A potential shortcoming of the proof of Theorem 1 is that the convergence time bounds it leads to tend to scale poorly in terms of the number of nodes $n$. We can overcome this shortcoming as illustrated in the following propositions. These results apply to a much narrower class of scenarios, but they tend to provide more effective bounds when they are applicable.

The first step is to introduce a precise notion of convergence time. Let $T(n, \epsilon, \{A^0, A^2, \ldots, \})$ denote the first time $k$ when

$$
\frac{\left\|x^k - \frac{\sum_{i=1}^n x^0_i}{n} 1\right\|}{\left\|x^0 - \frac{\sum_{i=1}^n x^0_i}{n} 1\right\|} \leq \epsilon.
$$

In other words, the convergence time is defined as the time until the deviation from the mean shrinks by a factor of $\epsilon$. The convergence time is a function of the desired accuracy $\epsilon$ and of the underlying sequence of matrices. In particular, we emphasize the dependence on the number of nodes, $n$. When the sequence of matrices is clear from context, we will simply write $T(n, \epsilon)$.

**Proposition 4.** Suppose

$$
x^{k+1} = A^k x^k
$$

where each $A^k_i$ is a doubly stochastic matrix. Then

$$
\left\|x^k - \frac{\sum_{i=1}^n x^0_i}{n} 1\right\|_2 \leq \left(\sup_{l=0, 1, 2, \ldots} \sigma_2 \left(A^l\right)\right)^k \left\|x^0 - \frac{\sum_{i=1}^n x^0_i}{n} 1\right\|_2,
$$

where $\sigma_2(A^l)$ denotes the second-largest singular value of the matrix $A^l$.

We skip the proof, which follows quickly from the definition of singular value. We adopt the slightly non-standard notation

$$
\lambda = \sup_{l \geq 0} \sigma_2 \left(A^l\right),
$$

so that the previous proposition can be conveniently restated as

$$
\left\|x^k - \frac{\sum_{i=1}^n x^0_i}{n} 1\right\| \leq \lambda^k \left\|x^0 - \frac{\sum_{i=1}^n x^0_i}{n} 1\right\|.
$$
Recalling that $\log(1/\lambda) \leq 1/(1 - \lambda)$, a consequence of this equation is that

$$T(n, \epsilon, \{A^0, A^1, \ldots\}) = O\left(\frac{1}{1 - \lambda} \ln \frac{1}{\epsilon}\right),$$

so the number $\lambda$ provides an upper bound on the convergence rate of distributed averaging.

In general, there is no guarantee that $\lambda < 1$, and the equations we have derived may be vacuous. Fortunately, it turns out that for the lazy Metropolis matrices on connected graphs, it is true that $\lambda < 1$, and furthermore, for many families of undirected graphs it is possible to give order-accurate estimates on $\lambda$, which translate into estimates of convergence time. This is captured in the following proposition. Note that all of these bounds should be interpreted as scaling laws, explaining how the convergence time increases as the network size $n$ increases, when the graphs all come from the same family.

**Proposition 5.** If each $A^k$ is the lazy Metropolis matrix on the...

1. ...path graph, then $T(n, \epsilon) = O\left(n^2 \log(1/\epsilon)\right)$.
2. ...2-dimensional grid, then $T(n, \epsilon) = O\left(n \log n \log(1/\epsilon)\right)$.
3. ...$k$-dimensional grid, then $T(n, \epsilon) = O\left(n^{1/k} \log n \log(1/\epsilon)\right)$.
4. ...star graph, then $T(n, \epsilon) = O\left(n^2 \log(1/\epsilon)\right)$.
5. ...two-star graph, then $T(n, \epsilon) = O\left(n \log n \log(1/\epsilon)\right)$.
6. ...complete graph, then $T(n, \epsilon) = O(1)$.
7. ...expander graph, then $T(n, \epsilon) = O(\log(1/\epsilon))$.
8. ...Erdős-Rényi random graph, then with high probability $T(n, \epsilon) = O(\log(1/\epsilon))$.
9. ...geometric random graph, then with high probability $T(n, \epsilon) = O(n \log n \log(1/\epsilon))$.
10. ...any connected undirected graph, then $T(n, \epsilon) = O\left(n^2 \log(1/\epsilon)\right)$.

**Sketch of the proof.** The spectral gap $1/(1 - \lambda)$ can be bounded by $O(H)$ where $H$ is the largest hitting time of the Markov chain whose probability transition matrix is the lazy Metropolis matrix $A^k$. We thus only need to bound hitting times on the graphs in question, and these are now standard exercises. For example, the fact that the hitting time on the path graph is quadratic in the number of nodes is essentially the main finding of the standard “gambler’s ruin” exercise. For corresponding results on 2-d and $k$-dimensional grids, please see [19]. Hitting times on star, two-star, and complete graphs are elementary exercises. The result for an expander graph is a consequence of Cheeger’s inequality; see Theorem 6.2.1 in [16]. For Erdős-Rényi graphs the result follows because such graphs are expanders with high probability; see [16]. For geometric random graphs a bound can be obtained by partitioning the unit square into appropriately-sized regions, thereby reducing to the case of a 2-d grid; see [20]. Finally the bound for connected graphs is from [18], [21].

Fig. 1 depicts examples of some of the graphs discussed in Proposition 5. Clearly the network structure affects the time it takes information to diffuse across the graph. For graphs such as the path or 2-d grid, the dependence on $n$ is intuitively related to the long time it takes information to spread across the network. For other graphs, such as stars, the dependence is due to the central node (i.e., the “hub”) becoming a bottleneck. For such graphs this dependence is strongly related to the fact that we have focused on the Metropolis scheme for designing the entries of the matrices $A^k$. Because the hub has a much higher degree than the other nodes, the resulting Metropolis updates lead to very small changes and hence slower convergence (i.e., $A^k$ is diagonally dominant); see Eq. (3). In general, for undirected graphs in which neighboring nodes may have very different degrees, it is know that faster rates can be achieved by using linear iterations of the form Eq. (2), where $A^k$ is optimized for the particular graph topology [22], [23]. However, unlike using the Metropolis weights—which can be implemented locally by...
Fig. 1. Examples of some graph families mentioned in Prop. 5. (a) Path graph with \( n = 5 \). (b) 2-d grid with \( n = 16 \). (c) Star graph with \( n = 6 \). (d) Two-star graph with \( n = 12 \).

having neighboring nodes exchange their degrees—determining the optimal matrices \( A^k \) involves solving a separate network-wide optimization problem; see [23] for a distributed approximation algorithm.

On the other hand, the algorithm is evidently fast on certain graphs. For the complete graph (where every node is directly connected to every other node, this is not surprising—since \( A = (1/n)11^T \), the average is computed exactly at every other node after a single iteration. Expander graphs can be seen as sparse approximations of the complete graph (sparse here is in the sense of having many fewer edges) which approximately preserve the spectrum, and hence the hitting time [24]. In applications where one has the option to design the network, expanders are particular practical interest since they allow fast rates of convergence—hence, few iterations—while also having relatively few links—so each iteration requires few transmissions and is thus fast to implement.

B. Worst-case scaling of distributed averaging

One might wonder about the worst-case complexity of average consensus: how long does it take to get close to the average on any graph? Initial bounds were exponential in the number of nodes [1]–[3], [25]. However, Proposition 5 tells us that this is at most \( O(n^2) \) using a Metropolis matrix. A recent result [18] shows that if all the nodes know an upper bound \( U \) on the total number of nodes which is reasonably accurate, this convergence time can be brought down by an order of magnitude. This is a consequence of the following theorem.

**Theorem 6.** Suppose each node in an undirected connected graph \( G \) implements the update

\[
\begin{align*}
w_i^{k+1} &= u_i^k + \frac{1}{2} \sum_{j \in N_i} \frac{u_j^k - u_i^k}{\max(d_i, d_j)}, \\
u_i^{k+1} &= w_i^{k+1} + \left(1 - \frac{2}{9U + 1}\right) \left(w_i^{k+1} - u_i^k\right),
\end{align*}
\]

(7)

where \( U \geq n \) and \( u^0 = w^0 \). Then

\[
\|w^k - \overline{w}\|^2 \leq 2 \left(1 - \frac{1}{9U}\right)^k \|w^0 - \overline{w}\|^2,
\]

where \( \overline{w} = (1/n) \sum_{i=1}^n w_i^0 \) is the initial average.

Thus if every node knows the upper bound \( U \), the above theorem tells us that convergence time until every element of the vector \( w^k \) is at most \( \epsilon \) away from the initial average \( \overline{w} \) is \( O(U \ln(1/\epsilon)) \). In the event that \( U \) is within a constant factor of \( n \), (e.g., \( n \leq U \leq 10n \)) this turns out to be linear in the number of nodes \( n \). One situation in which this is possible is if nodes precisely know the number of nodes in the network, in which case they can simply set \( U = n \). However, this scheme is also useful in a number of settings where the exact number of nodes in the system is not known (e.g., if nodes fail) as long as approximate knowledge of the total number of nodes is available.

Intuitively, Eq. (7) takes a lazy Metropolis update and accelerates it by adding an extrapolation term. Strategies of this form are known as over-relaxation in the numerical analysis literature [26] and as Nesterov acceleration in the optimization literature [27]. On a non-technical level, the extrapolation speeds up convergence by reducing
the inherent oscillations in the underlying sequence. A key feature, however, is that the degree of extrapolation must be carefully chosen, which is where knowledge of the bound $U$ is required. It is, at present, an open question whether any improvement on the quadratic convergence time of Proposition 5 is possible without such an additional assumption, or whether a linear convergence time scaling can be obtained for time-varying graphs.

III. DECENTRALIZED OPTIMIZATION OVER UNDIRECTED GRAPHS

We now shift our attention from distributed averaging back to the problem of optimization. We begin by describing the (centralized) subgradient method, which is one of the most basic algorithms used in convex optimization.

A. The subgradient method

To avoid confusion in the sequel when we discuss distributed optimization methods, here we consider a serial method for minimizing a convex function $h: \mathbb{R}^d \to \mathbb{R}$. A vector $g \in \mathbb{R}^d$ is called a subgradient of $h$ at the point $x$ if

$$h(y) \geq h(x) + g^T (y - x), \quad \text{for all } x, y \in \mathbb{R}^n.$$  

If the function $h$ is continuously differentiable, then $g = \nabla h(x)$ is the only subgradient at $x$. However, subgradients are also well-defined over the entire domain of a convex function, even if it is not continuously differentiable, and in general, there are multiple subgradients at points $x$ where the function $h$ is not differentiable.

The subgradient method for minimizing the function $h$ is defined as the iterate

$$u^{k+1} = u^k - \alpha^k g^k,$$  

where $g^k$ is a subgradient of the function $h$ at the point $u^k$.

There are standard results on the convergence of the subgradient method. Intuitively, the subgradient $g^k$ is always a descent direction at the point $u^k$, so a step in the direction of the subgradient pulls the iterate closer to the global minimizers provided the step-size is small enough. If it is hard to know what step-size to take, then one can always take a step-size sequence $\alpha^k$ that decays to zero at a suitable rate so that the method comes close to the minimizer after enough steps. This intuition is captured by the following theorem (see, e.g., [28] for a proof).

**Theorem 7.** Let $U^*$ be the set of minimizers of the function $h: \mathbb{R}^d \to \mathbb{R}$. Assume that (i) $h$ is convex, (ii) $U^*$ is nonempty, (iii) $\|g\| \leq L$ for all subgradients $g$ of the function $h(\cdot)$, and (iv) the nonnegative step-size sequence $\alpha^k$ is “summable but not square summable,” i.e.,

$$\sum_{k=0}^{\infty} \alpha^k = +\infty \quad \text{and} \quad \sum_{k=0}^{+\infty} [\alpha^k]^2 < \infty.$$  

Then, for any $u^* \in U^*$, we have

$$\lim_{k \to \infty} h \left( \frac{\sum_{i=0}^{k} \alpha^i u^i}{\sum_{i=0}^{k} \alpha_i} \right) = h(u^*).$$  

Furthermore, if Eq. (8) is run for $T$ steps under the above assumptions, the (constant) choice of stepsize $\alpha^k = 1/\sqrt{T}$ for $k = 0, \ldots, T - 1$ yields

$$h \left( \frac{\sum_{k=0}^{T-1} u^k}{T} \right) - h(u^*) \leq \frac{\|u^0 - u^*\|^2 + L^2}{2\sqrt{T}}.$$  

Subgradients have similar properties to gradients for convex functions, and similar to gradients, they are linear. (This follows directly from the definition in Eq. (8).) Thus, if $g_1$ is a subgradient of a function $f_1$ at $x$ and $g_2$ is a subgradient of $f_2$ at $x$, then $g_1 + g_2$ is a subgradient of $f_1 + f_2$ at $x$. 


B. Decentralizing the subgradient method

We now return to the problem of distributed optimization. To recap, we have \( n \) nodes, interconnected in a time-varying network capturing which pairs of nodes can exchange messages. For now, assume that these networks are all undirected. (This is relaxed in Section \[\text{V}\] which considers directed graphs.) Node \( i \) knows the convex function \( f_i : \mathbb{R}^m \rightarrow \mathbb{R} \) and the nodes would like to minimize the function

\[
f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)
\]

in a distributed way. If all the functions \( f_1(x), \ldots, f_n(x) \) were available at a single location, we could directly apply the subgradient method to their average \( f(x) \):

\[
u^{k+1} = u^k - \frac{1}{n} \sum_{i=1}^{n} \nabla f_i^k,
\]

where \( \nabla f_i^k \) is a subgradient of the function \( f_i(\cdot) \) at \( u^k \). Unfortunately, this is not a distributed method under our assumptions, since only node \( i \) knows the function \( f_i(\cdot) \), and thus only node \( i \) can compute a subgradient of \( f_i(\cdot) \).

A decentralized subgradient method solves this problem by interpolating between the subgradient method and an average consensus scheme from Section \[\text{III}\]. In this scheme, node \( i \) maintains the variable \( x_i^k \) which is updated as

\[
x_i^{k+1} = \sum_{j \in N_i^k} a_{ij}^k x_j^k - \alpha^k g_i^k,
\]

where \( g_i^k \) is the subgradient of \( f_i(\cdot) \) at \( x_i^k \). Note that this update is decentralized in the sense that node \( i \) only requires local information to execute it. In the case when \( f : \mathbb{R} \rightarrow \mathbb{R} \), the quantities \( x_j^k \) are scalars and we can write this as

\[
x^{k+1} = A x^k - \alpha^k g^k,
\]

where the vector \( x^k \in \mathbb{R}^n \) stacks up the \( x_i^k \) and \( g^k \in \mathbb{R}^n \) stacks up the \( g_i^k \). The weights \( a_{ij}^k \) should be chosen by each node in a distributed way. Later within this section, we will assume that the matrices \( A^k \) are doubly stochastic; perhaps the easiest way to achieve this is to use the Metropolis iteration of Eq. \[\text{3}\].

Intuitively, Eq. \[\text{10}\] pulls the value \( x_i^k \) at each node in two directions: one the one hand towards the minimizer (via the subgradient term) and on the other hand towards neighboring nodes (via the averaging term). The step-size \( \alpha^k \) should be chosen to decay to zero, so that in the limit the consensus term will prevail. However, if the rate at which it decays to zero is small enough, then under appropriate conditions consensus will be achieved on the global minimizer of \( f(\cdot) \).

We now turn to the analysis of Eq. \[\text{10}\]. For simplicity, we make the assumption that all the functions \( f_i(\cdot) \) are from \( \mathbb{R} \) to \( \mathbb{R} \); this simplifies the presentation but otherwise has no effect on the results. The same analysis extends in a straightforward manner to functions \( f_i : \mathbb{R}^d \rightarrow \mathbb{R} \) with \( d > 1 \) but requires more cumbersome notation.

**Theorem 8.** Let \( \mathcal{X}^* \) denote the set of minimizers of the function \( f \). We assume that: (i) each \( f_i \) is convex; (ii) \( \mathcal{X}^* \) is nonempty; (iii) each function \( f_i \) has the property that its subgradients at any point are bounded by the constant \( L \); (iv) the matrices \( A^k \) are doubly stochastic and there exists some \( \alpha > 0 \) such that the graph sequence \( [G_{A^0}]_\alpha, [G_{A^1}]_\alpha, [G_{A^2}]_\alpha, \ldots \) satisfies Assumption \[\text{I}\] and (v) the initial values \( x_i^0 \) are the same across all nodes (e.g., \( x_i^0 = 0 \)). Then:

1. If the step-sizes \( \alpha^k \) is “summable but not square summable,” i.e.,
   \[
   \sum_{k=0}^{\infty} \alpha^k = +\infty \quad \text{and} \quad \sum_{k=0}^{+\infty} \left[ \alpha^k \right]^2 < \infty,
   \]
   then for any \( x^* \in \mathcal{X}^* \), we have that for all \( i = 1, \ldots, n \),
   \[
   \lim_{t \to \infty} f \left( \frac{\sum_{l=0}^{t} \alpha^l x_i^l}{\sum_{l=0}^{t} \alpha^l} \right) = f(x^*).
   \]
2) If we run the protocol for \( T \) steps with (constant) step-size \( \alpha^k = 1/\sqrt{T} \), and with the notation \( y^i = (1/n) \sum_{i=1}^{n} x^i \), then we have that for all \( i = 1, \ldots, n \),

\[
f \left( \frac{\sum_{i=0}^{T-1} y^i}{T} \right) - f(x^*) \leq \frac{(y^0 - x^*)^2 + L^2}{2\sqrt{T}} + \frac{L^2}{\sqrt{T}} \tag{11}
\]

where \( \lambda \) is defined by Eq. (2).

We remark that the quantity \( (\sum_{i=0}^{T-1} y^i)/T \) on which the suboptimality bound is proved can be computed via an average consensus protocol after the protocol is finished if node \( i \) keeps track of \( (\sum_{i=0}^{T-1} x^i_i)/T \).

Comparing Theorems [7] and [8] and ignoring the similar terms involving the initial conditions, we see that the convergence bound gets multiplied by \( 1/(1 - \lambda) \). This term may be thought of as measuring the “price of decentralization” resulting from having knowledge of the objective function distributed throughout the network rather than available entirely at one place.

We can use Proposition [5] to translate this into concrete convergence times on various families of graphs, as the next result shows. For \( \epsilon > 0 \), let us define the \( \epsilon \)-convergence time to be the first time when

\[
f \left( \frac{\sum_{i=0}^{T-1} y^i}{T} \right) - f(x^*) \leq \epsilon.
\]

Naturally, the convergence time will depend on \( \epsilon \) and on the underlying sequence of matrices/graphs.

**Corollary 9.** Suppose all the hypotheses of Theorem [8] are satisfied, and suppose further that the weights \( \alpha_{ij}^k \) are the Metropolis weights defined in Eq. (3). Then the convergence time can be upper bounded as

\[
O \left( \frac{\max \left( (y^0 - x^*)^4, L^4 P_n^2 \right)}{\epsilon^2} \right),
\]

where if the graphs \( G^i \) are

1) ...path graphs, then \( P_n = O \left( n^2 \right) \);
2) ...2-dimensional grids, then \( P_n = O \left( \sqrt{n \log n} \right) \);
3) ...k-dimensional grids, then \( P_n = O \left( n^{1/k} \log n \right) \);
4) ...complete graphs, then \( P_n = O(1) \);
5) ...expander graphs, then \( P_n = O(1) \);
6) ...star graphs, then \( P_n = O \left( n^2 \right) \);
7) ...two-star graphs, then \( P_n = O \left( n^2 \right) \);
8) ...Erdős-Rényi random graphs, then \( P_n = O(1) \);
9) ...geometric random graphs, then \( P_n = O(n \log n) \);
10) ...any connected undirected graph, then \( P_n = O \left( n^2 \right) \).

These bounds follow immediately by putting together the upper bounds on \( 1/(1 - \lambda) \) discussed in Proposition [5] with Eq. (11).

We remark that it is possible to decrease the scaling from \( O(L^4 P_n^2) \) to \( O(L^2 P_n) \) in the above corollary if the constant \( L \), the type of the underlying graph (e.g., star graph, path graph) and the number of nodes \( n \) is known to all nodes. Indeed, this can be achieved by setting the stepsize \( \alpha^k = \beta/\sqrt{T} \) and using a hand-optimized \( \beta \) (which will depend on \( L, n \), as well as the kind of underlying graphs). We omit the details but this is very similar to the optimization done in (11).

We now turn to the proof of Theorem [8] We will need two preliminary lemmas covering some background in optimization.

**Lemma 10.** Suppose \( h : \mathbb{R}^m \rightarrow \mathbb{R} \) is a convex function such that \( h(\cdot) \) has subgradients \( g_x, g_y \) at the points \( x, y \), respectively, satisfying \( \|g_x\|_2 \leq L \) and \( \|g_y\|_2 \leq L \). Then

\[
|h(y) - h(x)| \leq L\|y - x\|_2
\]

**Proof.** On the one hand, we have by definition of subgradient

\[
h(y) \geq h(x) + g_x^T (y - x)
\]
so that by Cauchy-Schwarz
\[ h(y) - h(x) \geq -L \|y - x\|_2 \quad (12) \]
and on the other hand
\[ h(x) \geq h(y) + g_y^T (x - y) \]
so that
\[ h(x) - h(y) \geq -L \|x - y\|_2 \]
which we rearrange as
\[ h(y) - h(x) \leq L \|y - x\|_2. \quad (13) \]
Together Eq. (12) and Eq. (13) imply the lemma.

Our overall proof strategy is to view the distributed optimization protocol of Eq. (10) as a kind of perturbed consensus process. To that end, the next lemma extends our previous analysis of the consensus process to deal with perturbations.

**Lemma 11.** Suppose
\[ x^{k+1} = A^k x^k + \Delta^k, \]
where \( A^k \) are doubly stochastic matrices satisfying Assumption 7 and \( \Delta^k \in \mathbb{R}^n \) are perturbation vectors. Then:
1) If \( \sup_k \|\Delta^k\|_2 \leq L' \) then
\[ \|x^k - \frac{1}{n} x^0\|_2 \leq \lambda^k \|x^0 - \frac{1}{n} x^0\|_2 + \frac{L'}{1 - \lambda}, \]
where \( \lambda \) is from Eq. (4).
2) If \( \Delta^k \to 0 \), then \( x^k - (1^T x^k)/n \to 0 \).

**Proof.** For convenience, let us introduce the notation
\[ y^k = \frac{1}{n} x^k, \quad e^k = x^k - y^k 1, \quad m^k = \frac{1^T \Delta^k}{n}. \]
Since
\[ y^{k+1} = y^k + m^k \]
we have that
\[ e^{k+1} = A^k e^k + \Delta^k - m^k 1, \]
and therefore
\[ e^k = A^{k-1:0} e^0 + A^{k-1:1} (\Delta^0 - m^0 1) \]
\[ + \cdots + A^{k-1:k-1} (\Delta_{k-2} - m^{k-2} 1) + (\Delta_{k-1} - m^{k-1} 1). \]
Now using the fact that the vectors \( e^0 \) and \( \Delta^j - m^j 1 \) have mean zero, by Proposition 4 we have
\[ \|e^k\|_2 \leq \lambda^k \|e^0\|_2 + \sum_{j=0}^{k-1} \lambda^{k-1-j} \|\Delta^j\|_2. \quad (14) \]
This equation immediately implies the first claim of the lemma.

Now consider the second claim. We define
\[ L^k_{\text{first-half}} = \sup_{0 \leq j < k/2} \|\Delta^j\|_2 \]
\[ L^k_{\text{second-half}} = \sup_{k/2 \leq j \leq k} \|\Delta^j\|_2 \]
Since \( \Delta^k \to 0 \), we have
\[ \sup_k L^k_{\text{first-half}} < \infty, \quad \lim_k L^k_{\text{second-half}} = 0. \quad (15) \]
Now Eq. (14) implies
\[ \|e^k\|_2^2 \leq \lambda^{k-1}\|e^0\|_2^2 + \lambda^{k/2}\frac{L_{\text{first-half}}^k}{1 - \lambda} + \frac{L_{\text{second-half}}^k}{1 - \lambda}. \]

Combining this with Eq. (15), we have that \( \|e^k\|_2 \to 0 \) and this proves the second claim of the lemma.

Proof of Theorem 8. Recall that we are assuming, for simplicity, that the functions \( f_i \) are from \( \mathbb{R} \) to \( \mathbb{R} \). As before, let \( y^k \) be the average of the entries of the vector \( x^k \in \mathbb{R}^n \), i.e.,
\[ y^k = \frac{1}{n} x^k. \]

Since the matrices \( A^k \) are doubly stochastic, \( 1^T A^k = 1^T \) so that
\[ y^{k+1} = y^k - \alpha^k \frac{1^T g^k}{n}. \]

Now for any \( x^* \in \mathcal{X}^* \), we have
\[ (y^{k+1} - x^*)^2 \leq (y^k - x^*)^2 + \left[ \alpha^k \right]^2 L^2 - 2\alpha^k \frac{1}{n} \sum_{i=1}^{n} g^k_i (y^k - x^*). \] (16)

Furthermore, for each \( i = 1, \ldots, n \),
\[ g^k_i (y^k - x^*) = g^k_i (x^*_i - x^* + y^k - x^*_i) \]
\[ = g^k_i (x^*_i - x^*) + g^k_i (y^k - x^*_i) \]
\[ \geq f_i(x^*_i) - f_i(x^*) - L |y^k - x^*_i| \]
\[ \geq f_i(y^k) - f_i(x^*) - 2L |y^k - x^*_i| \]

where the first inequality uses a rearrangement of the definition of the subgradient and the last inequality uses Lemma 10. Plugging this into Eq. (16), we obtain
\[ (y^{k+1} - x^*)^2 \leq (y^k - x^*)^2 + \left[ \alpha^k \right]^2 L^2 - 2\alpha^k \frac{1}{n} \sum_{i=1}^{n} g^k_i (y^k - x^*) \]

or
\[ 2\alpha^k (f(y^k) - f(x^*)) \leq (y^k - x^*)^2 - (y^{k+1} - x^*)^2 \]
\[ + \left[ \alpha^k \right]^2 L^2 + 2L\alpha^k \frac{1}{n} \sum_{i=1}^{n} |y^k - x^*_i|. \]

We can sum this up to obtain
\[ 2 \sum_{l=0}^{t} \alpha^l \left( f(y^l) - f(x^*) \right) \]
\[ \leq (y^0 - x^*)^2 - (y^{l+1} - x^*)^2 \]
\[ + L^2 \sum_{l=0}^{t} \left[ \alpha^l \right]^2 + 2L \sum_{l=0}^{t} \alpha^l \frac{1}{n} \sum_{i=1}^{n} |y^l - x^*_i|, \]
which in turn implies
\[
\sum_{i=0}^{t} \alpha_l y^l - f(x^*)
\]
\[
\leq \frac{(y^0 - x^*)^2 + L^2 \sum_{i=0}^{t} [\alpha_l]^2 + 2L \sum_{i=0}^{t} \alpha_l (1/n) \sum_{i=1}^{n} |y^l - x_i^l|}{2 \sum_{i=0}^{t} \alpha_l}
\]
\[
= \frac{(y^0 - x^*)^2 + L^2 \sum_{i=0}^{t} [\alpha_l]^2}{2 \sum_{i=0}^{t} \alpha_l} + \frac{2L \sum_{i=0}^{t} \alpha_l (1/n) \sum_{i=1}^{n} |y^l - x_i^l|}{2 \sum_{i=0}^{t} \alpha_l}.
\]

We now turn to the first claim of the theorem statement. The first term on the right-hand side goes to zero because its numerator is bounded while its denominator is unbounded (due to the assumption that the step-size is summable but not square summable). For the second term, we view $-\alpha^k y^k$ as the perturbation $\Delta^k$ in Lemma 2 to obtain that $x^l - y^l \to 0$, and in particular $x_i^l - y_i^l \to 0$ for each $i$. It follows that the Cesàro sum (which is exactly the second term on the right-hand side) must go to zero as well. We have thus shown that
\[
f \left( \frac{\sum_{i=0}^{t} \alpha_l y^l}{\sum_{i=0}^{t} \alpha_l} \right) - f(x^*) \to 0.
\]
Putting this together with Lemma 11, which implies that $x_i^l - y_i^l \to 0$ for all $i$, we complete the proof of the first claim.

For the second claim, using (i) Lemma 11 (ii) the assumption that the initial conditions are the same across all nodes, and (iii) the inequalities $\|z\|_1 \leq \sqrt{n} \|z\|_2$ for vectors $z \in \mathbb{R}^n$, we have the bound
\[
f \left( \frac{\sum_{i=0}^{t} \alpha_l y^l}{\sum_{i=0}^{t} \alpha_l} \right) - f(x^*)
\]
\[
\leq \frac{(y^0 - x^*)^2 + L^2 \sum_{i=0}^{t} \alpha_l^2/1/\sqrt{T} (1/n) \left( L \sqrt{n}/\sqrt{T} (1/\lambda) \right) \left( \sqrt{T} (1/\lambda) \right)}{2 \sqrt{T}}
\]
\[
= \frac{(y^0 - x^*)^2 + L^2}{2 \sqrt{T}} + \frac{L^2}{\sqrt{T} (1 - \lambda)},
\]
which proves the second claim of the theorem.

\[\square\]

C. Improved scaling with the number of nodes

The results of Corollary 9 improve upon those reported in [11], [29], [30]. A natural question is whether it is possible to further improve the scalings even further. In particular, one might wonder how the worst-case convergence time of distributed optimization scales with the number of nodes in the network. In general, this question is open. Partial progress was made in [18], [21], where, under the assumption that all nodes know an order-accurate bound on the total number of nodes in the network, it was shown that we can use the linear time convergence of average consensus described in Theorem 6 to obtain a corresponding convergence time for distributed optimization when the underlying graph is fixed and undirected. Specifically, [18], [21] consider the following update rule
\[
y_i^{k+1} = x_i^k + \frac{1}{2} \sum_{j \in N_i} y_j^k - x_j^k \max(d_i, d_j) - \beta g_i^k
\]
\[
z_i^{k+1} = y_i^k - \beta g_i^k
\]
\[
x_i^{k+1} = y_i^{k+1} + \left( 1 - \frac{2}{9U + 1} \right) \left( y_i^{k+1} - z_i^{k+1} \right),
\]
where $g_i^k$ is a subgradient of the function $f_i(\cdot)$ at the point $y_i^k$. As in Section III here the number $U$ is an upper bound on the number of nodes known to each individual node, and it is assumed that $U$ is within a constant factor
of the true number of nodes, i.e., \( n \leq U \leq cn \) for some constant \( c \) (not depending on \( n, U \) or any other problem parameters).

By relying on Theorem 6, it is shown in [18], [21] that the corresponding time until this scheme (followed by a round of averaging) is \( \epsilon \) close to consensus on a minimizer of \( \frac{1}{n} \sum_{i=1}^{n} f_i(\cdot) \) is \( O(\log n + n/\epsilon^2) \). It is an open question at present whether a similar convergence time can be achieved over time-varying graphs or without knowledge of the upper bound \( U \).

**D. The strongly convex case**

The error decrease of \( 1/\sqrt{T} \) with the number of iterations \( T \) is, in general, the best possible rate for dimension-independent convex optimization [31]. However, under the stronger assumption that the underlying functions \( f_i(\cdot) \) are strongly convex with Lipschitz-continuous gradients, gradient descent will converge geometrically. Until recently, however, there were no corresponding distributed protocols with a geometric rate.

Significant progress on this issue was first made in [32], where, over fixed undirected graphs, the following scheme was proposed:

\[
x^{k+2} = (I + W)x^{k+1} - \tilde{W}x^k - \alpha \left[ \nabla f(x^{k+1}) - \nabla f(x^k) \right],
\]

with the initialization

\[
x^1 = Wx^0 - \alpha \nabla f(x^0).
\]

Here, for simplicity, we continue with the assumption that the functions \( f_i(\cdot) \) are from \( \mathbb{R} \) to \( \mathbb{R} \). The matrices \( W \) and \( \tilde{W} \) are two different appropriately chosen symmetric stochastic matrices compatible with the underlying graph; for example, \( W \) might be taken to be the Metropolis matrix with \( \tilde{W} = (I + W)/2 \). It was shown in [32] that this scheme, called EXTRA, drives all nodes to the global optimal at a geometric rate under natural technical assumptions.

The EXTRA update of Eq. (19) looks rather mysterious at first glance. Moreover, without a good intuition for what it is doing, it is difficult to extend it to time-varying or directed graphs (the original proof in [32] only covered fixed graphs). Progress on this question was made in [33], which besides providing a geometrically convergent method in the time-varying/directed cases, gave an intuitive explanation for why EXTRA works. Indeed, [33] observed that the scheme

\[
\begin{align*}
x^{k+1} &= W^kx^k - \alpha y^k \\
y^{k+1} &= W^ky^k + \nabla f(x^{k+1}) - \nabla f(x^k)
\end{align*}
\]

is a special case of the EXTRA update of Eq. (19). Here the initialization \( x^0 \) can be arbitrary, while \( y^0 = \nabla f(x^0) \). The matrices \( W^k \) are doubly stochastic. Moreover, Eq. (20) has a natural interpretation.

Indeed, the second line of Eq. (20) is a tracking recursion: \( y^k \) tracks the time-varying gradient average \( 1^T \nabla f(x^k)/n \).

Indeed, observe that, by the double stochasticity of \( W^k \), we have that

\[
1^Ty^k/n = 1^T \nabla f(x^k)/n.
\]

In other words, the vector \( y^k \) has the same average as the average gradient. Moreover, it is easy to see that if \( x^k \to \bar{x} \), then \( y^k \to \nabla f(\bar{x}) \); this is due to the “consensus effect” of repeated multiplications by \( W^k \). Such recursions for tracking were studied in [34].

While the second line of Eq. (20) tracks the average gradient, the first line of Eq. (20) performs a gradient step as if \( y^k \) was the exact gradient direction. The method can be naturally analyzed using methods for approximate gradient descent. It was shown in [33] that this method converges to the global optimizers geometrically under the same assumptions as EXTRA, even when the graphs are time-varying; further, the complexity of reaching an \( \epsilon \) neighborhood of the optimal solution is polynomial in direction.

We conclude by remarking that there is quite a bit of related work in the literature. Indeed, the idea to use “two-layered” scheme as in Eq. (20) originates from [35]–[39]. Furthermore, improved analysis of convergence rates over an undirected graph is available in [40] and [41].
IV. DIRECTED GRAPHS

A. Distributed averaging over directed graphs

We have seen in Section II that over time-varying undirected graphs, the lazy Metropolis update results in consensus on the initial average. In this section, we ask whether this is possible over a sequence of directed graphs.

By way of motivation, we remark that many of the applications of distributed optimization in engineering involve directed graphs. For example, in wireless networks the communication radius of a node is a function of its broadcasting power; if nodes do not all transmit at the same power level, communications will naturally be directed. Any distributed optimization protocol meant to work in wireless networks must be prepared to deal with unidirectional communications.

Unfortunately, it turns out that there is no direct analogue of the lazy Metropolis method for average consensus over directed graphs. In fact, if we consider deterministic protocols where, at each step, nodes broadcast information to their neighbors and then update their states based on the messages they have received, then it can be proven no such protocol can result in average consensus; see [42]. We thus need to make an additional assumption to solve the average consensus problem over directed graphs.

A standard assumption in the field is that every node always knows its out-degree. In other words, whenever a node broadcasts a message it knows how many other nodes are within listening range. In practice, this can be accomplished in practice via a two-level scheme, wherein nodes broadcast hello-messages at an identical and high power level, while the remainder of the messages are transmitted at lower power levels. The initial exchange of hello-messages provides estimates of distance to neighboring nodes, allowing each node to see how many listeners it has as a function of its transmission power. Alternatively, the out-degrees can be estimated in a distributed manner using linear iterations [43].

Under this assumption, it turns out that average consensus is indeed possible and may be accomplished via the following iteration,

\[
\begin{align*}
    x_{i}^{k+1} &= \sum_{j \in N_{i}^{\text{in},k}} \frac{x_{j}^{k}}{d_{j}^{\text{out}}}, \\
    y_{i}^{k+1} &= \sum_{j \in N_{i}^{\text{in},k}} \frac{y_{j}^{k}}{d_{j}^{\text{out}}},
\end{align*}
\]  

initialized at an arbitrary \(x^{0}\) and \(y^{0} = 1\). This is known as the Push Sum iteration; it was introduced in [44], and studied recently in [45], [46].

The name Push Sum derives from the nature of the distributed implementation of Eq. (21). Observe that Eq. (21) may be implemented with one-directional communication. Specifically, every node \(i\) transmits (or broadcasts) the values \(x_{i}^{k} / d_{i}^{\text{out}}\) and \(y_{i}^{k} / d_{i}^{\text{out}}\) to its out-neighbors. After these transmissions, each node has the information it needs to perform the update (21), which involves summing the pushed values. In contrast, the algorithms for undirected graphs described in the previous sections required that each node \(i\) send a message to all of its neighbors and receive a message from each neighbor. Protocols of this sort are known as “push-pull” in the distributed computing literature because the transmission of a message from node \(i\) to node \(j\) (the “push”) implies that \(i\) also expects to receive a message from \(j\) (the “pull”).

Our next theorem, which is the main result of this subsection, tells us that Push-Sum works. For this result, we define matrices \(A^{k}\) as follows:

\[
a^{k}_{i,j} = \begin{cases} 
\frac{1}{d_{i}^{\text{in}}} & \text{if } j \in N_{i}^{\text{in},k}, \\
0 & \text{else}.
\end{cases}
\]

Theorem 12. Suppose the sequence of graphs \(G_{A^{0}}, G_{A^{1}}, G_{A^{2}}, \ldots\) satisfies Assumption[7] Then for each \(i = 1, \ldots, n\),

\[
\lim_{k \to \infty} \frac{x_{i}^{k}}{y_{i}^{k}} = \frac{\sum_{j=1}^{n} x_{j}^{0}}{n}.
\]

It is somewhat remarkable that the convergence to the average happens for the ratios \(x_{i}^{k} / y_{i}^{k}\). Adopting the notation \(a./b\) for the element-wise ratio of two vectors \(a\) and \(b\), the above theorem may be restated as

\[
\lim_{k \to \infty} x_{i}^{k} \cdot / y_{i}^{k} = \left( \frac{\sum_{j=1}^{n} x_{j}^{0}}{n} \right) 1.
\]
We now turn to the proof of the theorem. Using the matrices \(A^k\) as defined in Eq. (22), the iterations in Eq. (21) may be written as

\[
x^{k+1} = A^k x^k, \quad y^{k+1} = A^k y^k.
\]

Observe that \(A^k\) is column stochastic by design, i.e.,

\[
1^T A^k = 1^T.
\]

As a consequence of this, the sums of \(x^k\) and \(y^k\) are preserved, i.e.,

\[
\sum_{i=1}^{n} x^k_i = \sum_{i=1}^{n} x^0_i, \quad \sum_{i=1}^{n} y^k_i = \sum_{i=1}^{n} y^0_i = n. \tag{23}
\]

For our proof, we will need to use the fact that the vector \(y^k\) remains strictly positive and bounded away from zero in each entry. This is shown in the following lemma.

**Lemma 13.** For all \(i, k\), \(y^k_i \geq 1/(n^{2nB})\).

**Proof.** By Assumption 1 every node has a self-loop, so we have that

\[
y^{k+1}_i \geq \frac{1}{n} y^k_i
\]

and consequently the lemma is true for \(k = 1, \ldots, nB - 1\). Let \(lnB\) be the largest multiple of \(nB\) which is at most \(k\). If \(k > nB - 1\) then

\[
y^k = A^{k-nB} A^{nB-1:0} 1.
\]

By Lemma 2 the matrix \(A^{nB-1:0}\) is the transpose product of stochastic matrices satisfying Assumption 1 and consequently each of its entries is at least \(\alpha^{nB}\) (where \(\alpha = 1/n\)) by Lemma 2. Thus

\[
\begin{bmatrix} A^{nB-1:0} 1 \end{bmatrix}_i \geq \left( \frac{1}{n} \right)^{nB} \text{ for all } i = 1, \ldots, n.
\]

Applying Eq. (24) to the last \(k - nB\) steps now proves the lemma. \(\square\)

With this lemma in mind, we can give a proof of Theorem 12 that is essentially a quick reduction to the result already obtained in Theorem 1.

**Proof of Theorem 12.** Let us introduce the notation \(z^k_i = x^k_i/y^k_i\). Then \(x^k_i = z^k_i y^k_i\), and therefore, we can rewrite the Push Sum update as

\[
\begin{align*}
\sum_{j=1}^{n} [W^k]_{ij} z^k_j y^k_j,
\end{align*}
\]

or

\[
z^{k+1}_i = \sum_{j=1}^{m} [W^k]_{ij} (y^{k+1}_j)^{-1} z^k_j y^k_j,
\]

where the last step used the fact that \(y^k_i \neq 0\), which follows from Lemma 13. Therefore, defining

\[
P^k = \left( \text{diag}(y^{k+1}) \right)^{-1} W^k \text{diag}(y^k),
\]

we have that

\[
z^{k+1} = P^k z^k.
\]

Moreover, \(P^k\) is stochastic since

\[
P^k 1 = \left( \text{diag}(y^{k+1}) \right)^{-1} W^k \text{diag}(y^k) 1
\]

\[
= \left( \text{diag}(y^{k+1}) \right)^{-1} W^k y^k
\]

\[
= \left( \text{diag}(y^{k+1}) \right)^{-1} y^{k+1}
\]

\[
= 1.
\]
We have thus written the Push Sum update as an “ordinary” consensus update after a change of coordinates. However, to apply Theorem 14 about the convergence of the basic consensus process, we need to lower bound the entries of $P^k$, which we proceed to do next.

Indeed, as a consequence of the definition of $P^k$, if we choose $\alpha$ to be some fixed number such that
\[
\alpha \leq \left( \max_i y_i \right)^{-1} \min_{(i,j)} |W^i|_{ij} \left( \min_i y_i \right)
\]
always holds, then the sequence of graphs $G_{[P^0]}$, $G_{[P^1]}$, $G_{[P^2]}$, \ldots will satisfy Assumption 1. To find an $\alpha$ that satisfies this condition, we make use of the fact that $1/(n^{2nB}) \leq y_i \leq n$, which is a consequence of Eq. (23) and Lemma 13. It follows that the choice $\alpha = (1/n) \cdot (1/n) \cdot 1/(n^{2nB})$ suffices. Thus, we can apply Theorem 1 and obtain that $z^k$ converges to a multiple of the all-ones vector.

It remains to show that the final limit point is the initial average. Let $z_{\infty}$ be the ultimate limit of each $z_i^k$. Then for each $k = 0, 1, 2, \ldots$,
\[
\begin{align*}
\lim_{k \to \infty} \left( z_{\infty} - \frac{\sum_{i=1}^n x_i^k}{n} \right) &= \lim_{k \to \infty} \frac{\sum_{i=1}^n (z_{\infty} - z_i^k) y_i^k}{n} = 0,
\end{align*}
\]
where the last equality used that each $y_i^k$ is a positive number upper bounded by $n$. Finally, appealing to the first relation in Eq. (23) we complete the proof of the theorem.

\[\square\]

**B. Push Sum-based subgradient method**

Suppose now that every agent $i$ has a (scalar) convex objective function $f_i(\cdot)$, and the system objective is to minimize $f(x) = (1/n) \sum_{i=1}^n f_i(x)$. We next describe a distributed subgradient method for determining a minimizer of $f$ using the push-sum algorithm. Every node $i$ maintains scalar variables $x_i^k$, $y_i^k$, $w_i^k$, and updates them according to the following rules: for all $k \geq 0$ and all $i = 1, \ldots, n$,
\[
\begin{align*}
w_i^{k+1} &= \sum_{j \in N_{in,k}} \frac{x_j^k}{d_{ij}^k}, \\
y_i^{k+1} &= \sum_{j \in N_{im,k}} \frac{y_j^k}{d_{ij}^k}, \\
z_i^{k+1} &= \frac{w_i^{k+1}}{y_i^{k+1}}, \\
x_i^{k+1} &= w_i^{k+1} - \alpha^{k+1} y_i^{k+1}.
\end{align*}
\]
where $g_i^{k+1}$ is a subgradient of the function $f_i(z)$ at $z = z_i^{k+1}$. The method is initiated with an arbitrary vector $x_i^0 \in \mathbb{R}$ at node $i$, and with $y_i^0 = 1$ for all $i$. The push-sum updates steer the vectors $z_i^{k+1}$ toward each other in order to converge to a common point, while the subgradients in the updates of $x_i^{k+1}$ drive this common point to lie in the set of minimizers of the objective function $f$.

In the next theorem, we establish the convergence properties of the subgradient method of Eq. (25).

**Theorem 14.** Let $\mathcal{X}^*$ be the set of minimizers of the function $f$. We assume that: (i) each $f_i$ is convex; (ii) $\mathcal{X}^*$ is nonempty; (iii) each function $f_i$ has the property that its subgradients at any point are bounded by a constant $L$; (iv) the graph sequence $G_{\mathcal{X}^0}, G_{\mathcal{X}^1}, G_{\mathcal{X}^2}, \ldots$ satisfies Assumption 7. Then:
1) If the stepsizes $\alpha^1, \alpha^2, \ldots$ are positive, non-increasing, and satisfy the conditions
\[ \sum_{k=1}^{\infty} \alpha^k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} [\alpha^k]^2 < \infty, \]
then the distributed subgradient method of Eq. (25) converges asymptotically:
\[ \lim_{k \to \infty} z^k_i = x^* \quad \text{for all } i \text{ and for some } x^* \in X^*. \]

2) If $\alpha^k = 1/\sqrt{k}$ for $k \geq 1$ and every node $i$ maintains the variable $z^k_i \in \mathbb{R}$ initialized at $k = 0$ with any $z^0_i \in \mathbb{R}$ and updated by
\[ z^{k+1}_i = \frac{\alpha^{k+1} z^{k+1}_i + S^{k} z^k_i}{S^{k+1}} \quad \text{for } k \geq 0, \]
where $S^0 = 0$ and $S^k = \sum_{s=0}^{k-1} \alpha^{s+1}$ for $k \geq 1$, then for all $k \geq 1$, $i = 1, \ldots, n$, and any $x^* \in X^*$,
\[ f \left( z^{k+1}_i \right) - f^* \leq \frac{n}{2} \frac{|\bar{x}(0) - x^*|}{\sqrt{k+1}} + \frac{L^2 (1 + \ln(k + 1))}{2n \sqrt{k+1}} \]
\[ + \frac{24L \sum_{j=1}^{n} |x^0_j|}{\delta(1-\lambda) \sqrt{k+1}} + \frac{24L^2 (1 + \ln k)}{\delta(1-\lambda) \sqrt{k+1}}, \]
where $f^*$ is the optimal value of the problem, i.e., $f^* = \min_{z \in \mathbb{R}} f(z)$, and $\bar{x}(0) = \frac{1}{n} \sum_{i=1}^{n} x^0_i$. The scalars $\delta$ and $\lambda$ are functions of the graph sequence $G_{A^0}, G_{A^1}, G_{A^2}, \ldots$; in particular,
\[ \delta \geq \frac{1}{nB} \quad \text{and} \quad \lambda \leq \left( 1 - \frac{1}{nB} \right)^{1/(nB)}. \]

We note that the lower bounds on $\delta$ and $\lambda$ can be refined when some additional structure is imposed on the underlying time-varying graphs. The results of Theorem 14 and their proofs can be found in [47] for a more general case when $f_i$ are defined over $\mathbb{R}^d$ with $d \geq 1$. A better convergence rate can be obtained under the additional assumption that the objective functions $f_i$ are strongly convex; see [48].

The first work to have employed Push Sum distributed averaging within a distributed optimization methods is [9], and it was further investigated in [49]–[51]. This work focused on static graphs, and it has been proposed as an alternative to the algorithm based on synchronized distributed averaging over undirected graphs in order to avoid deadlocks and synchronization issues, among others. This work also described a distributed method based on Push Sum for multi-agent optimization problems with constraints by using Nesterov’s dual-averaging approach. This Push Sum consensus-based algorithm has been extended to the subgradient-push algorithm in [47], [48] that can deal with convex optimization problems over time-varying directed graphs. More recently, the paper [52] has extended the Push Sum algorithm to a larger class of distributed algorithms that are applicable to nonconvex objectives, convex constraint sets, and time-varying graphs.

References [53], [54] combine EXTRA with the Push Sum approach to produce the DEXTRA (Directed Extra-Push) algorithm for optimization over a directed graph. It has been shown that DEXTRA converges at a geometric (R-linear) rate for a strongly convex objective function, but it requires a careful stepsize selection. It has been noted in [53] that the feasible region of stepsizes which guarantees this convergence rate can be empty in some cases.

V. EXTENSIONS AND OTHER WORK ON DISTRIBUTED OPTIMIZATION

We discuss here some extension as well as other algorithms for minimizing the average sum $f(\cdot) = \frac{1}{n} \sum_{i=1}^{n} f_i(\cdot)$ in a distributed manner.
A. Extensions

1) Per-agent constraints: When the nodes have a common convex and closed constraint set (known to each node) \( X \subset \mathbb{R}^m \), the algorithms discussed in Section III easily extend to handle the simple set constraints by performing projections on the set \( X \). For example, the distributed subgradient method in Eq. (10) can be modified to the following update rule:

\[
x_i^{k+1} = \Pi_X \left[ \sum_{j \in N_i^k} a_{ij} x_j^k - \alpha^k g_i^k \right],
\]

where \( \Pi_X[\cdot] \) is the Euclidean projection on the set \( X \). The subgradient \( g_i^k \) of the function \( f_i \) can be evaluated at the past iterate \( x_i^k \) or at the point \( \sum_{j \in N_i^k} a_{ij} x_j^k \). Since the projection mapping \( \Pi_X[\cdot] \) is non-expansive (i.e., \( \| \Pi_X[x] - \Pi_X[y] \|_2 \leq \|x - y\|_2 \) for all \( x, y \)), the convergence properties of the algorithm with projections remain the same as that of the algorithm without projections.

A more complicated case arises when the constrain set is given as the intersection of individual node constraints, i.e., \( X = \cap_{i=1}^n X_i \), where each \( X_i \) is a convex closed set and only known to node \( i \). In this case, the node \( i \) update in Eq. (26) is modified by replacing \( \Pi_X[\cdot] \) with \( \Pi_{X_i}[\cdot] \). The convergence analysis in this case is more involved, as it requires some form of error bound results (also known as set regularity conditions). This algorithm, including random set-selections, have been studied in [53]–[58] for synchronous updates over time-varying graphs and for gossip-based asynchronous updates over a static graph. A variant of this algorithm (using the Laplacian formulation of the consensus problem) for distributed optimization with distributed constraints in noisy networks has been studied in [59], [60].

2) Asynchronous vs synchronous computations: All the algorithms we discussed so far have been synchronous in the sense that all nodes update at the same time and also use the same stepsize \( \alpha^k \) at iteration \( k \). To accommodate the asynchronous updates and, also, allow that agents use different stepsizes, one may resort to gossip or broadcast communications, where a random link is activated for communication (gossip) or a random node is activated to broadcast its information to neighbors. Consensus algorithms implemented in a network using a gossip-based or a broadcast-based communications have been studied in [61]–[64], while a different consensus algorithm (so called push-sum method) have been considered in [44], [45]. A nonlinear gossip method is investigated in [65], while a survey paper [66] provides a detailed account of gossip algorithms for distributed averaging and their applications to signal processing in sensor networks.

B. Additional Work on Distributed Optimization

A distributed algorithm preserving an optimality condition at every iteration has been proposed in [67]. Distributed convex optimization algorithms for weight-balanced directed graphs have been investigated in continuous-time [68].

A different type of a distributed algorithm for convex optimization has been proposed in [69], where each agent keeps an estimate for all agents’ decisions. This algorithm solves a problem where the agents have to minimize a global cost function \( f(x_1, \ldots, x_m) \) while each agent \( i \) can control only its variable \( x_i \). The algorithm of [69] has been recently extended to the online optimization setting in [70]–[72]. Distributed algorithms using augmented Lagrangian approach with gossip-type communications has been studied in [12], while accelerated versions of distributed gradient methods have been proposed and studied in [73]. A consensus-based algorithm for solving problems with a separable constraint structure and the use of primal-dual distributed methods have been studied in [34], [74], [75], while a distributed primal-dual approach with perturbations have been explored in [76]. Work in [77] provides algorithms for centralized and distributed convex optimization from control perspective, while [78] considers an event-triggered distributed optimization for sensor networks. In [79], a distributed simplex algorithm has been developed for linear programming problems, while a Newton-Raphson consensus-based method has been proposed in [80] for distributed convex problems.

Although our discussion has mainly focused on studying asymptotic rates of convergence of iterative distributed optimization methods, in [81] it was shown that a related approach based on consensus can solve general constrained abstract optimization problems in a finite number of iterations.

All of the work mentioned above relies on the use of state-independent weights, i.e., the weights that do not depend on the agents’ iterates. A consensus-based algorithm employing state-dependent weights has been proposed and analyzed in [82].
Another popular distributed approach for a static network is through the use of Alternate Direction Multiplier Method (ADMM). This method is based on an equivalent formulation of the consensus constraints. Unlike consensus-based (sub)-gradient method, which operates in the space of the primal-variables, the ADMM solves a corresponding Lagrangian dual problem (obtained by relaxing the equality constraints that are associated with consensus requirement). Just as any dual method, the ADMM is applicable to problems where the structure of the objective functions $f_i$ is simple enough so that the ADMM updates can be executed efficiently. The algorithm has potential of solving the problem with geometric convergence rate, which requires global knowledge of some parameters including eigenvalues of a weight matrix associated with the graph. A recent survey on the ADMM and its various applications is given in [83]. The first work to address the development of distributed ADMM over a network is [84]–[86], and it has been investigated in [87], while its linear rate has been shown in [88]. In [89] the ADMM with linearization has been proposed for special composite optimization problems over graphs.

The work in [35], [36] utilizes an adapt-then-combine (ATC) strategy [90], [91] of dynamic weighted-average consensus approach [92] to develop a distribute algorithm, termed Aug-DGM algorithm. This algorithm can be used over static directed or undirected graphs (but requires doubly stochastic matrix). The most interesting aspect of the Aug-DGM algorithm is that it can produce convergent iterates even when different agents use different (constant) stepsizes.

Simultaneously and independently, the idea of tracking the gradient averages through the use of consensus has been proposed in [35] for convex unconstrained problems and in [37] for non-convex problems with convex constraints. The work in [37]–[39] develops a large class of distributed algorithms, referred to as NEXT, which utilizes various “function-surrogate modules” thus providing a great flexibility in its use and rendering a new class of algorithms that subsumes many of the existing distributed algorithms. The work in [38], [39] and in [36] have also been proposed independently, with the former preceding the latter. The algorithm framework of [37]–[39] is applicable to nonconvex problems with convex constraint sets over time-varying graphs, but requires the use of doubly stochastic matrices. This assumption was recently removed in [52] by using column-stochastic matrices, which are more general than the degree-based column-stochastic matrices of the push-sum method. Simultaneously and independently, the papers [39] and [93] have appeared to treat nonconvex problems over graphs. The work in [93] proposes and analyzes a distributed gradient method based on the push-sum consensus in deterministic and stochastic setting for unconstrained problems.

The idea of using a consensus process to track gradients has also been recently used in [33] to develop a distributed algorithm, referred to as DIGing, with a geometric convergence rate over time-varying graphs. This is the first paper to establish such a rate for a consensus-based algorithms for convex optimization over time-varying graphs. We note that the algorithm uses a fixed stepsize, and the rate result is applicable to the problems with a strongly convex smooth objective function.

VI. CONCLUSION AND OPEN PROBLEMS

We have discussed distributed optimization methods for minimizing the average of the nodes’ objectives over graphs. We have considered undirected and directed time varying graphs, and computational models for solving consensus problem in such graphs. Then, we have discussed distributed optimization algorithms that combine optimization techniques with distributed averaging algorithms. We have also discussed extensions of the consensus-based approaches and other distributed optimization algorithms.

In terms of algorithm scalability with the number $n$ of nodes, at present, it is an open question whether any improvement on the quadratic convergence time of Proposition 5 is possible without an additional assumption about the knowledge of $n$, which was made in Theorem 6 to show a linear scaling with $n$. Also, it is not known whether a linear convergence-time scaling can be obtained for time-varying graphs.

Another question for future research is the implementation of distributed algorithms with lower communication requirements. In particular, even broadcast based communications can be expensive, in terms of the power needed to broadcast in some sensor networks. A question is how to implement distributed algorithms with fewer communications, and what trade-offs are involved in such implementations. Some initial investigations along these lines were presented in [94] in the context of stochastic optimization, where progressively more time is spent calculating gradients between each round of communication as the number of iterations progresses. There remains much further work to be done along these lines.
Finally, we remark that although there are well-understood lower bounds on the number of iterations required to achieve an \( \epsilon \)-optimal solution in the context of centralized convex optimization\cite{31, 95}, much less is understood about the fundamental limits of distributed optimization. Although bounds on the number of iterations for centralized algorithms carry over directly to synchronous distributed algorithms, since any distributed algorithm can always be emulated on a centralized processor, these results do not provide insight into how much communication is fundamentally required to reach consensus on an \( \epsilon \)-optimal solution. In communication-constrained settings (e.g., where network links have very low bandwidth), it remains an open question as to how many iterations may be required, and a related line of questioning would be to understand when there may be tradeoffs between communication and computation (e.g., to reach an \( \epsilon \)-optimal solution there may be algorithms which require significant computation and lower communication, or vice versa).

REFERENCES

[1] J. Tsitsiklis, D. Bertsekas, and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[2] J. Tsitsiklis, “Problems in decentralized decision making and computation,” Ph.D. dissertation, M.I.T., Nov. 1984.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[4] M. Rabbat and R. Nowak, “Distributed optimization in sensor networks,” in *Proc. ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, Berkeley, CA, USA, Apr. 2004, pp. 20–27.

[5] I. Schizas, G. Giannakis, S. Roumeliotis, and A. Ribeiro, “Consensus in ad hoc WSNs with noisy links - part ii: Distributed estimation and smoothing of random signals,” *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1650–1666, Apr. 2008.

[6] F. Cattivelli and A. Sayed, “Diffusion LMS strategies for distributed estimation,” *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[7] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[8] S. Ram, A. Nedic, and V. Veeravalli, “A new class of distributed optimization algorithms: Application to regression of distributed data,” *Optimization Methods and Software*, vol. 27, no. 1, pp. 71–88, 2009.

[9] K. Tsianos, S. Lawlor, and M. Rabbat, “Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning,” in 50th Allerton Conference on Communication, Control, and Computing, 2012.

[10] J. Chen and A. H. Sayed, “Diffusion adaptation strategies for distributed optimization and learning over networks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, Aug. 2012.

[11] J. Duchi, A. Agarwal, and M. Wainwright, “Dual averaging for distributed optimization: Convergence analysis and network scaling,” *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[12] D. Jakovetić, J. Xavier, and J. Moura, “Cooperative convex optimization in networked systems: Augmented lagrangian algorithms with directed gossip communication,” *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3889–3902, 2011.

[13] J. Cortés, S. Martínez, and F. Bullo, “Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions,” *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, Aug. 2006.

[14] Y. Cao, W. Yu, W. Ren, and G. Chen, “An overview of recent progress in the study of distributed multi-agent coordination,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[15] M. Zavlanos, A. Ribeiro, and G. Pappas, “Network integrity in mobile robotic networks,” *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 3–18, Jan. 2013.

[16] R. Durrett, *Random Graph Dynamics*. Cambridge University Press, 2007.

[17] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003.

[18] A. Olshevsky, “Linear time average consensus on fixed graphs,” in *Proceedings of NeCSys, the 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2015.

[19] A. K. Chandra, P. Raghavan, W. Ruzzo, R. Smolensky, and P. Tiwari, “The electrical resistance of a graph captures its commute and cover times,” *Computational Complexity*, vol. 6, no. 4, pp. 312–340, 1996.

[20] C. Avin and G. Ercal, “On the cover time and mixing time of random geometric graphs,” *Theoretical Computer Science*, vol. 380, no. 4, pp. 2–22, 2007.

[21] A. Olshevsky, “Linear time average consensus on fixed graphs and implications for decentralized optimization and multi-agent control,” 2014, to appear in *SIAM Journal on Control and Optimization*, available at http://arxiv.org/abs/1411.4186.

[22] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems and Control Letters*, vol. 53, pp. 65–78, 2004.

[23] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.

[24] D. Spielman, “Graphs, vectors, and matrices,” *Bulletin of the American Mathematical Society*, vol. 54, no. 1, pp. 45–61, Jan. 2017.

[25] A. Jadbabaie, J. Lin, and A. Morse, “Coordination of groups of mobile autonomous agents using nearest neighbor rules,” *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[26] R. Varga, *Matrix iterative analysis*. Springer, 2000.

[27] Y. Nesterov, “A method of solving a convex optimization problem with convergence rate \( o(1/k^2) \),” *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[28] B. Polyak, “A general method for solving extremum problems,” *Soviet Mathematical Doklady*, vol. 8, no. 3, pp. 593–597, 1967.
K. Tsianos and M. Rabbat, “Distributed consensus and optimization under communication delays,” in *Proceedings of the 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 2055–2060.

J. Xu, “Augmented distributed optimization for networked systems,” Ph.D. dissertation, School of Electrical and Electronic Engineering, Nanyang Technological University, 2016.

P. D. Lorenzo and G. Scutari, “Distributed nonconvex optimization over networks,” in *Proceedings of IEEE International Conference on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 2015)*, Dec. 13–16, 2015, Cancun, Mexico, 2015.

——, “Distributed nonconvex optimization over time-varying networks,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 16)*, March 20–25, 2016, Shanghai, China, 2016.

——, “NEXT: In-network nonconvex optimization,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

G. Qu and N. Li, “Accelerated distributed nesterov gradient descent,” preprint. Available at [https://arxiv.org/abs/1705.07176](https://arxiv.org/abs/1705.07176).

——, “Harnessing smoothness to accelerate distributed optimization,” 2017, to appear in *IEEE Transactions on Control of Network Systems*.

J. M. Hendrickx and J. N. Tsitsiklis, “Fundamental limitations for anonymous distributed systems with broadcast communications,” in *Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing*, 2015.

T. Charalambous, M. Rabbat, M. Johansson, and C. Hadjicostis, “Distributed finite-time computation of digraph parameters: Left-eigenvector, out-degree, and spectrum,” *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 137–148, Jun. 2016.

D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information,” in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003, pp. 482–491.

F. Benezit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, “Weighted gossip: distributed averaging using non-doubly stochastic matrices,” in *Proceedings of the 2010 IEEE International Symposium on Information Theory*, Jun. 2010.

A. Dominguez-Garcia and C. Hadjicostis, “Distributed strategies for average consensus in directed graphs,” in *Proceedings of the IEEE Conference on Decision and Control*, Dec 2011.

A. Nedić and A. Olshhevsky, “Distributed optimization over time-varying directed graphs,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

——, “Stochastic gradient-push for strongly convex functions on time-varying directed graphs,” *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3936 – 3947, 2016.

K. Tsianos, S. Lawlor, and M. Rabbat, “Push-sum distributed dual averaging for convex optimization,” in *Proceedings of the IEEE Conference on Decision and Control*, 2012.

K. Tsianos and M. Rabbat, “Distributed consensus and optimization under communication delays,” in *Proc. of Allerton Conference on Communication, Control, and Computing*, 2011, pp. 974–982.

K. Tsianos. “The role of the network in distributed optimization algorithms: Convergence rates, scalability, communication / computation tradeoffs and communication delays,” Ph.D. dissertation, McGill University, Dept. of Electrical and Computer Engineering, 2013.

Y. Sun, G. Scutari, and D. Palomar, “Distributed nonconvex multiagent optimization over time-varying networks,” in *Proc. of the Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, Nov. 2016.

C. Xi and U. Khan, “On the linear convergence of distributed optimization over directed graphs,” 2015, available on arXiv at [http://arxiv.org/abs/1510.02149](http://arxiv.org/abs/1510.02149).

J. Zeng and W. Yin, “Extrapush for convex smooth decentralized optimization over directed networks,” 2015, available on arXiv at [http://arxiv.org/abs/1511.02942](http://arxiv.org/abs/1511.02942).

A. Nedić, A. Ozdaglar, and P. Parrilo, “Constrained consensus and optimization in multi-agent networks,” *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922 –938, April 2010.

S. Lee, “Optimization over networks: Efficient algorithms and analysis,” Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 2013.

S. Lee and A. Nedić, “Distributed random projection algorithm for convex optimization,” *IEEE Journal on Selected Topics in Signal Processing*, vol. 48, no. 6, pp. 988–1001, 2012.

——, “Asynchronous gossip-based random projection algorithms over networks,” *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 953–968, 2016.

K. Srivastava, A. Nedić, and D. Stipanović, “Distributed constrained optimization over noisy networks,” in *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 1945 –1950.

K. Srivastava and A. Nedić, “Distributed asynchronous constrained stochastic optimization,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 772–790, 2011.

S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Gossip algorithms: Design, analysis, and applications,” in *Proceedings of IEEE INFOCOM*, vol. 3, 2005, pp. 1653–1664.

T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, “Broadcast gossip algorithms: Design and analysis for consensus,” in *Proceedings of the 47th IEEE Conference on Decision and Control*, 2008, pp. 4843–4848.
S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *SIAM Journal on Control and Optimization*, vol. 56, no. 1, pp. 1168–1199, 2008.

T. Tatarenko and B. Touri, “Non-convex distributed optimization,” 2016, preprint. Available at http://arxiv.org/abs/1512.00895.

K. Tsianos, S. Lawlor, and M. Rabbat, “Communication/computation tradeoffs in consensus-based distributed optimization,” in *Proc. Advances in Neural Information Processing Systems*, Lake Tahoe, USA, Dec. 2012, pp. 1943–1951.

Y. Nesterov, *Introductory Lectures on Convex Optimization*. Springer, 2003.