Distributed Optimization with Sheaf Homological Constraints

Jakob Hansen
Department of Mathematics
University of Pennsylvania
Philadelphia, Pennsylvania
jhansen@math.upenn.edu

Robert Ghrist
Department of Mathematics
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, Pennsylvania
ghrist@math.upenn.edu

***Abstract*—In this paper we introduce a new class of local linear operators on graphs, the *sheaf Laplacians*, which provide drop-in replacements for graph Laplacians in distributed algorithms. These operators can enforce more general constraints on data distributed in a network than those given by the graph Laplacian. The constraints for such optimization problems can be framed in the context of *sheaf cohomology*, leading to a description of this framework as distributed optimization with homological constraints. We formulate a representative problem, elucidate its solution with sheaf Laplacians, and give illustrative examples of potential applications.**

***Index Terms*—optimization, distributed systems, spectral graph theory, algebraic topology, cellular sheaf**

I. Introduction

Distributed optimization is a broad field with applications in machine learning, systems control, sensor networks, and many other areas. Algorithms for distributed optimization typically act over a network modeled as a graph, with each node both performing local optimization over its own parameters and communicating with its neighbors to share parameters. Typically the parameter spaces for each node are identical, and the goal is to have each node converge to the same optimum of a joint objective function given by some combination of objective functions available to each node.

In our formulation, the overall goal of optimizing a sum of local objective functions remains, but we relax the assumption that each node has the same parameter space, and nodes are not required to communicate their entire vector of parameters to every neighbor. Such network communications structures have not typically been considered, although they have been technically possible to construct by hand. Our contribution gives an interpretation of certain types of generalized constraints for data on a network in terms of mathematical structures called *cellular sheaves* [1] and their associated Laplacian operators [2]. Cellular sheaves represent local systems of linear constraints for data on a network, and the sheaf Laplacians recently introduced by the authors allow immediate translation of these local constraints into a local linear operator which can be immediately substituted into any distributed algorithm that makes use of graph Laplacians.

This work was supported with funding from DARPA and the AFRL under agreement number FA8650-18-2-7840.

II. Distributed Optimization

Distributed optimization has an illustrious history spanning a diversity of models and approaches. Some of the earliest work is due to Tsitsiklis and collaborators, studying asynchronous methods where all agents know the entire objective function [3]. More recent work has applied other notions from convex optimization such as primal-dual methods and the alternating direction method of multipliers [4], [5]. Other work has studied situations where agents have only partial knowledge of the objective function [6], [7]. Algorithms for these problems often involve a combination of a local optimization process (e.g., gradient descent) and a global consensus process (e.g., graph diffusion). These processes can be implemented and integrated in different ways, and the ultimate goals and assumptions of these algorithms vary. The function to be optimized may be known to all agents in the network, or it may be broken up into a sum of pieces, each of which is known to one agent. Agents may have only stochastic approximations to the true cost function. Communication may happen between every pair of neighbors at every time, or gossip-type algorithms may be used. The common thread, however, is that agents are able to cooperate to find an optimum using information distributed throughout the network. We will follow here a framework established by Wang and Elia [8], [9], which due to its general formulation in terms of Lagrangian dynamics and graph Laplacian constraints is readily adaptable to the case of sheaf Laplacians. Our description will focus on the unconstrained problem, but it is of course possible to apply this approach to problems with local convex constraints.

In this framework, the agents are connected according to a graph G with a set of nodes V. Each node $v \in V$ is able to compute a locally known convex, smooth objective function $f_v : \mathbb{R}^n \to \mathbb{R}$ and its gradients. The global objective function is $f(x) = \sum_{v \in V} f_v(x)$ for $x \in \mathbb{R}^n$. Since each node has its own local parameter in the state space $\mathbb{R}^n$, we reformulate the problem to make this redundancy explicit:

$$\begin{aligned} &\min \sum_{V \in V} f_v(x_v) \\ &\text{s.t. } x_u = x_v \; \forall u, v. \end{aligned} \tag{1}$$

This problem is equivalent to the original problem in the sense that a minimizer of (1) consists of one copy of the minimizer of f for each vertex. This problem is further converted into one reflecting the network structure by only including the equality constraints corresponding to edges $u \sim v$. Letting $\mathbf{x} = \text{vec}(\{x_v\}_{v \in V})$, we can cast this constraint in terms of the graph Laplacian:

$$(L \otimes I)\mathbf{x} = 0.$$

(Throughout, we will use bold variables to denote "vectors of vectors" created by concatenating local state variables.)

We make one more adjustment to the problem before producing a distributed algorithm. Although this is not strictly necessary, we add the term $\mathbf{x}^T(L \otimes I)\mathbf{x}$ to the objective function, to improve stability and convergence of the resulting algorithm while leaving the minimum and minimizer undisturbed. Our problem then becomes

$$\begin{aligned} &\min_{\mathbf{x}} \sum_{V \in V} f_v(x_v) + \mathbf{x}^T(L \otimes I)\mathbf{x} \\ &\text{s.t. } (L \otimes I)\mathbf{x} = 0. \end{aligned} \tag{2}$$

The Lagrangian corresponding to this problem is

$$\mathcal{L}(\mathbf{x}, \mathbf{z}) = \sum_{V \in V} f_v(x_v) + \mathbf{x}^T(L \otimes I)\mathbf{x} + \mathbf{z}^T(L \otimes I)\mathbf{x}.$$

From it, we can build a dynamical system whose stationary points are those points satisfying the Karush-Kuhn-Tucker (KKT) conditions for (2):

$$\begin{aligned} \dot{\mathbf{x}} &= -\frac{\partial}{\partial \mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{z}) = -\nabla \mathbf{f}(\mathbf{x}) - 2(L \otimes I)\mathbf{x} - (L \otimes I)\mathbf{z} \\ \dot{\mathbf{z}} &= \frac{\partial}{\partial \mathbf{z}}\mathcal{L}(\mathbf{x}, \mathbf{z}) = (L \otimes I)\mathbf{x}, \end{aligned} \tag{3}$$

where $\mathbf{f}(\mathbf{x}) = \sum_{v \in V} f_v(x_v)$. These are the saddle-point dynamics associated with $\mathcal{L}$; we may think of them as a sort of primal-dual evolution, since we perform gradient descent in the primal variable $\mathbf{x}$ and gradient ascent in the dual variable $\mathbf{z}$.

Saddle-point dynamics for functions convex in one argument and concave in the other are well studied. In particular, we have the following result from [10]:

Proposition II.1 (Corollary 4.5 in [10])**.** *Let $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be a continuously differentiable function which is globally convex-concave and linear in its second argument, such that for every saddle point (x^*, z^*) of F, if $F(x, z^*) = F(x^*, z^*)$, then (x, z^*) is also a saddle point of F. Then the set of saddle points of F is globably asymptotically stable under the saddle-point dynamics and each trajectory converges to a point.*

When the f_v are smooth and convex, the function $\mathcal{L}(\mathbf{x}, \mathbf{z})$ satisfies these conditions, so every trajectory of (3) converges to a point satisfying the KKT conditions for (2). Further, these dynamics are locally implementable: to compute $\dot{x}_v$ and $\dot{z}_v$, the agent at node v only needs to know the values of x_u and z_u for $u \sim v$.

This continuous-time system may then be discretized, most commonly with an Euler-type rule, yielding a discrete-time primal-dual optimization algorithm. The appropriate step size for an Euler discretization will depend on the eigenvalues of L and the Hessians of the f_v.

Extensions of this framework are possible, including implementing convex constraints on the variables x_v and the use of nonsmooth objective functions. For the sake of exposition, we focus on the simplest case. This framework is also far from the only approach to distributed optimization. Many approaches focus on a discrete-time evolution from the beginning, and some use only primal or only dual variables. Our use of the framework introduced by Wang and Elia is motivated by the ease with which it may be adapted to more general distributed problems.

III. Cellular Sheaves

Sheaves are a structure arising in algebraic topology and geometry, where they specify ways that data may be attached to a space. They control the ways that locally specified data may be patched together to globally consistent information. Cellular sheaves are a discrete, computable version of these structures amenable to real-world implementation. In the context of networks, cellular sheaves mediate the relationship between local constraints for data and the global integration of those constraints.

For our purposes, graphs are undirected, oriented, and loop-free, although there may be multiple edges between two vertices.

Definition III.1. Let $G = (V, E)$ be a graph. A *cellular sheaf* of vector spaces $\mathcal{F}$ over G is given by the following data:

- A vector space $\mathcal{F}(v)$ for each vertex v of G
- A vector space $\mathcal{F}(e)$ for each edge e of G
- A linear map $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ for each incident vertex-edge pair $v \trianglelefteq e$.

The vector spaces $\mathcal{F}(v)$ and $\mathcal{F}(e)$ are called the *stalks* over v and e, and the linear map $\mathcal{F}_{v \trianglelefteq e}$ is called a *restriction map*.

A sheaf first specifies places for data over a graph to reside. Data associated with a vertex v lives inside the stalk $\mathcal{F}(v)$. Similarly, if we associate data to edges, it lives in the edge stalks $\mathcal{F}(e)$. The restriction maps of the sheaf then describe local linear constraints for data over the graph. Specifically, over an edge $e = u \sim v$, the sheaf imposes the constraint $\mathcal{F}_{u \trianglelefteq e} x_u = \mathcal{F}_{v \trianglelefteq e} x_v$ for $x_u \in \mathcal{F}(u)$ and $x_v \in \mathcal{F}(v)$.

To formalize these constraints, we define the vector spaces of *cochains* associated with a cellular sheaf. These are the spaces

$$C^0(G; \mathcal{F}) = \bigoplus_{v \in V} \mathcal{F}(v) \quad C^1(G; \mathcal{F}) = \bigoplus_{e \in E} \mathcal{F}(e).$$

We think of $C^0(G; \mathcal{F})$, the space of 0-cochains, as a space of signals over vertices, and similarly the space of 1-cochains $C^1(G; \mathcal{F})$ as a space of signals over edges.

The local consistency constraints for data in the stalks of $\mathcal{F}$ assemble to global conditions on signals. We say that $\mathbf{x} =$

$\text{vec}(\{x_u\}_{u \in V(G)}) \in C^0(G;\mathcal{F})$ is a *global section* of $\mathcal{F}$ if it satisfies all local consistency constraints. That is, it must satisfy $\mathcal{F}_{u \trianglelefteq e} x_u = \mathcal{F}_{v \trianglelefteq e} x_v$ for every edge $e = u \sim v$. The global sections of $\mathcal{F}$ are globally consistent assignments of data to the vertices of G. The global sections of $\mathcal{F}$ are also called the degree-0 cohomology of $\mathcal{F}$, and are denoted $H^0(G;\mathcal{F})$.

Example. The simplest cellular sheaf is the *constant sheaf* with stalk a given vector space V. This is the sheaf with every stalk equal to V, and every restriction map the identity. We denote this sheaf $\underline{V}$. The global sections of $\underline{V}$ are the locally constant V-valued functions on G, i.e., those which are constant on every connected component of G.

A. Sheaf Laplacians

Naturally associated with a sheaf $\mathcal{F}$ on an (oriented, undirected) graph $G = (V, E)$ is a linear map $\delta : C^0(G;\mathcal{F}) \to C^1(G;\mathcal{F})$. This is called the *coboundary map*, and has the important role of computing global sections of $\mathcal{F}$. On an oriented edge $e = u \to v$, the coboundary map is defined by

$$(\delta \mathbf{x})_e = \mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u.$$

If we represent δ as a matrix with block structure corresponding to the direct sum decompositions of C^0 and C^1, it is block row sparse: each block row has only two nonzero blocks.

The coboundary map of a sheaf is analogous to the signed incidence matrix of a graph. Indeed, the coboundary map of the constant sheaf $\underline{\mathbb{R}}$ can be identified with the signed incidence matrix of the underlying graph. Since incidence matrices of graphs can be used to construct the graph Laplacian, it is natural, then, to build a sheaf Laplacian out of the sheaf coboundary map.

Definition III.2. Let $\mathcal{F}$ be a cellular sheaf on a graph G. The *sheaf Laplacian* of $\mathcal{F}$ is the map $L_{\mathcal{F}} : C^0(G;\mathcal{F}) \to C^0(G;\mathcal{F})$ given by $L_{\mathcal{F}} = \delta^* \delta$.

The sheaf Laplacian has a block structure induced by the block structure of δ. The diagonal block corresponding to a vertex v is $L_{v,v} = \sum_{v \trianglelefteq e} \mathcal{F}^*_{v \trianglelefteq e} \mathcal{F}_{v \trianglelefteq e}$, while the off-diagonal block corresponding to a pair of vertices u, v joined by an edge e is $L_{u,v} = -\mathcal{F}^*_{u \trianglelefteq e} \mathcal{F}_{v \trianglelefteq e}$. When applied to a vector $x \in C^0(G;\mathcal{F})$, we have $(L_{\mathcal{F}} x)_v = \sum_{u,v \trianglelefteq e \in E} \mathcal{F}^*_{v \trianglelefteq e} (\mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u)$.

Some facts about the Laplacian are immediate from its definition: it is a positive semidefinite matrix with $\ker L_{\mathcal{F}} = \ker \delta = H^0(G;\mathcal{F})$. Since the coboundary map of the constant sheaf is identical to the incidence matrix of its underlying graph, the sheaf Laplacian of the constant sheaf $\underline{\mathbb{R}}$ is the same as the graph Laplacian. Somewhat more generally, the sheaf Laplacian $L_{\underline{\mathbb{R}}^n}$ is equal to $L_G \otimes I_{n \times n}$. Note also that the sheaf Laplacian does not depend on the choice of orientation of G, just as the graph Laplacian does not.

We can use the sheaf Laplacian to give a quantitative measure of consistency or smoothness for elements of $C^0(G;\mathcal{F})$. The most consistent 0-cochains are those which are global sections, but we might also wish to consider cochains which are close to being consistent. The Laplacian quadratic form

$$\mathbf{x}^T L_{\mathcal{F}} \mathbf{x} = \sum_{e=u \sim v \in E} \| \mathcal{F}_{u \trianglelefteq e} x_u - \mathcal{F}_{v \trianglelefteq e} x_v \|^2$$

gives a convenient measure of the total edgewise discrepancy of a cochain $\mathbf{x} \in C^0(G;\mathcal{F})$. The behavior of this quadratic form is, of course, closely related to the spectrum of $L_{\mathcal{F}}$. We might call the eigenvector corresponding to the smallest nonzero eigenvalue of $L_{\mathcal{F}}$ the cochain closest to being a global section of $\mathcal{F}$, or perhaps the smoothest non-"constant" signal with respect to the structure specified by $\mathcal{F}$.

Sheaf Laplacians are a broad class of linear operators on networks. In many situations where graph Laplacians are used, sheaf Laplacians can serve as a drop-in substitute to add more general classes of constraints.

B. Some Examples of Cellular Sheaves

We give here two examples of cellular sheaves that may arise in engineering applications. The first is a general construction applicable to sensor networks.

Example (The sheaf of functions on a space)**.** Suppose we have a collection of sensors $\mathcal{S}$ which observe subsets U_i of some space X. That is, there is some function $f : X \to \mathbb{R}^n$, and each sensor s_i can measure $f|_{U_i} : U_i \to \mathbb{R}^n$. The sensors naturally have a network structure determined by overlaps of their domains: $s_i \sim s_j$ if and only if $U_i \cap U_j \neq \varnothing$. This leads naturally to a cellular sheaf on the sensor network. The stalk over a sensor is $\mathcal{F}(s_i) = \{f : U_i \to \mathbb{R}^n\}$, while the stalk over the edge between sensors s_i and s_j is $\mathcal{F}(s_i \sim s_j) = \{f : U_i \cap U_j \to \mathbb{R}^n\}$. The restriction map $\mathcal{F}(s_i) \to \mathcal{F}(s_i \sim s_j)$ is given by restriction of functions: $(f_i : U_i \to \mathbb{R}^n) \mapsto (f_i|_{U_i \cap U_j} : U_i \cap U_j \to \mathbb{R}^n)$.

The global sections of this sheaf (under the assumption of full sensor coverage of X) correspond precisely to the functions $X \to \mathbb{R}^n$. The vertex stalks encode the space of possible local observations for each sensor, while the constraints associated to each edge require that the observations of each pair of sensors agree where they overlap. In mathematical terminology, this is the sheaf of functions on X subordinate to the cover $\{U_i\}$.

When X is a discrete set, these spaces of functions are simply finite-dimensional vector spaces, and thus the construction of the sheaf Laplacian is simple. Restriction is simply an orthogonal projection, and its adjoint is inclusion or extension by zero.

Our second example is somewhat more concrete, focusing on the space of admissible trajectories of a set of coupled linear systems.

Example (Linear control systems)**.** Suppose we have a collection of discrete-time linear time-invariant systems Σ_i with states x_i, inputs u_i, and outputs y_i, given by evolution equations

$$\begin{aligned} x_i[t+1] &= A_i x[t] + B_i u_i[t] \\ y_i[t] &= C_i x[t] + D_i u[t], \end{aligned}$$

which are coupled via linear connections between inputs and outputs:

$$U_{ij}u_i[t] = Y_{ji}y_j[t]. \tag{4}$$

We may think of the maps U_{ij} and Y_{ji} as coordinate projections, so that only a subset of the inputs to system i are coupled to a subset of the outputs of system j, but this restriction is not necessary. These equations can be seen as a cellular sheaf $\mathcal{M}$. The underlying graph G has one vertex for each system Σ_i, and an edge between any two systems which have a coupling. The stalk over vertex i is the vector space of permissible trajectories of the system Σ_i: if the state space is n_i dimensional and the input space m_i dimensional, this space is the subspace of $(\mathbb{R}^{n_i} \oplus \mathbb{R}^{m_i})^\infty$ satisfying the state evolution equation for Σ_i. These vector spaces are infinite dimensional, but we will make a restriction down to finite-dimensional stalks shortly. The stalk over an edge $i \sim j$ is the space $(\operatorname{im} Y_{ij} \oplus \operatorname{im} U_{ij})^\infty = (\operatorname{im} U_{ji} \oplus \operatorname{im} Y_{ji})^\infty$. It represents the space in which the system couplings are defined at each time step. The restriction map $\mathcal{M}_{i \trianglelefteq (i\sim j)}$ is given by the formula

$$(\mathcal{M}_{i \trianglelefteq (i\sim j)}(x_i, u_i))[t] = (Y_{ij}(C_i x_i[t] + D_i u_i[t]), U_{ij}u_i[t]).$$

The restriction map $\mathcal{M}_{j \trianglelefteq (i\sim j)}$ is the same but with the order of factors switched, i.e.:

$$(\mathcal{M}_{j \trianglelefteq (i\sim j)}(x_j, u_j))[t] = (U_{ji}u_j[t], (Y_{ji}(C_j x_j[t] + D_j u_j[t])).$$

This means that the constraint imposed by the sheaf over the edge $i \sim j$ is precisely the coupling constraint (4). As a result, the global sections of $\mathcal{M}$ correspond precisely to trajectories of the coupled system.

Many modifications may be made to this sheaf. We can impose a finite time horizon by simply truncating the stalks to finite products of the state and coupling spaces. Rather than set the vertex stalks to subspaces of the state and input trajectory space, we can let $\mathcal{M}(i) = (\mathbb{R}^{n_i} \oplus \mathbb{R}^{m_i})^\infty$, and add a new vertex-edge pair constraining sections to be admissible trajectories. To do this, add a new vertex i' with stalk $\mathcal{M}(i') = 0$. The edge stalk is $\mathcal{M}(i \sim i') = (\mathbb{R}^{n_i})^\infty$. The restriction map from i' must be the zero map, and the restriction map from i is given by $(\mathcal{M}_{i \trianglelefteq (i\sim i')}(x_i, u_i))[t] = x_i[t+1] - A_i x_i[t] - B_i u_i[t]$.

Note also that the structure of this system leaves the possibility that trajectories are underdetermined: not all of the inputs of a given system may be coupled to anything. We may think of this as the system posessing a free input that may therefore be connected to a controller.

IV. Distributed Optimization with Sheaf Laplacians

Cellular sheaves and their Laplacians immediately suggest a generalized version of the distributed optimization problem considered in Section II. The simplest distributed optimization problem supported on a sheaf $\mathcal{F}$ over a graph G with vertex set V is of the form

$$\begin{aligned} &\min \sum_{v\in V} f_v(x_v) \\ &\text{s.t. } \mathbf{x} \in H^0(G;\mathcal{F}), \end{aligned}$$

where $f_v : \mathcal{F}(v) \to \mathbb{R}$ is a convex function for each $v \in V$.

If $\mathcal{F}$ is the constant sheaf with stalk $\mathbb{R}^n$, this problem becomes the standard distributed optimization problem on G, as $H^0(G;\underline{\mathbb{R}^n})$ is simply the space of constant $\mathbb{R}^n$-valued functions on vertices of G. Since $H^0(G;\mathcal{F})$ is simply $\ker L_\mathcal{F}$, the constraint $\mathbf{x} \in H^0(G;\mathcal{F})$ is equivalent to $L_\mathcal{F}\mathbf{x} = 0$. Thus any algorithm for distributed optimization that begins from the reformulation of a global consensus constraint as a constraint involving the graph Laplacian can be immediately applied to this problem of optimization with homological constraints.

To implement this in the Lagrangian saddle point dynamics framework, we rewrite the problem as a problem with the same minimizer

$$\begin{aligned} &\min \sum_{v\in V} f_v(x_v) + \mathbf{x}^T L_\mathcal{F}\mathbf{x} \\ &\text{s.t. } L_\mathcal{F}\mathbf{x} = 0. \end{aligned}$$

This problem is formally identical to the optimization problem discussed in Section II. As a result, the saddle point dynamics

$$\begin{aligned} \dot{\mathbf{x}} &= -\nabla \mathbf{f}(\mathbf{x}) - 2L_\mathcal{F}\mathbf{x} - L_\mathcal{F}\mathbf{z} \\ \dot{\mathbf{z}} &= L_\mathcal{F}\mathbf{x} \end{aligned}$$

converge under the same assumptions on the f_v to a point satisfying the KKT conditions. The locality of these dynamics is manifest when we write them vertexwise:

$$\begin{aligned} \dot{x}_v = &-\nabla f_v(x_v) - 2\sum_{\substack{v \trianglelefteq e \\ u \trianglelefteq e}} \mathcal{F}^*_{v\trianglelefteq e}(\mathcal{F}_{v\trianglelefteq e}x_v - \mathcal{F}_{u\trianglelefteq e}x_u) \\ &- \sum_{\substack{v \trianglelefteq e \\ u \trianglelefteq e}} \mathcal{F}^*_{v\trianglelefteq e}(\mathcal{F}_{v\trianglelefteq e}z_v - \mathcal{F}_{u\trianglelefteq e}z_u) \\ \dot{z}_v = &\sum_{\substack{v \trianglelefteq e \\ u \trianglelefteq e}} \mathcal{F}^*_{v\trianglelefteq e}(\mathcal{F}_{v\trianglelefteq e}x_v - \mathcal{F}_{u\trianglelefteq e}x_u) \end{aligned}$$

To compute its local gradient, each node v need only know the values $\mathcal{F}_{u\trianglelefteq e}x_u$ for each edge $e = u \sim v$. Translation to a discrete-time algorithm proceeds in the same way as for the classical problem, via Euler's method.

A. Related Work

The problem of distributed optimization with local linear constraints has been considered by Nassif et al [11] and Hua et al [12]. Both papers consider a formulation focused on a distributed online learning problem, with individual nodes learning local regression coefficients from streaming data. Under their model, the regression coefficients x_v corresponding to nodes in certain clusters C_k are required to satisfy affine relationships $\sum_{v\in C_k} A_{vk}x_v = b_k$. This is similar to (and indeed more general than) the constraints we consider, where each edge of the graph implements a linear constraint $\mathcal{F}_{v\trianglelefteq e}x_v = \mathcal{F}_{u\trianglelefteq e}x_u$.

Our approach differs in several ways. First, the introduction of the sheaf Laplacian makes the imposition of pairwise linear constraints between local parameters formally identical

to the standard consensus-based situation. There is no need to design an entirely new algorithm to approach the problem with sheaf homological constraints. Further, the similarity to standard Laplacian-based algorithms makes adaptation to new situations and problems simpler. For instance, the analysis in both [11] and [12] focuses on the case where the objective functions are least mean squares cost functions for linear regression-like problems, whereas the analysis here immediately applies to any smooth convex objective. Finally, the distributed algorithm proposed in [11] formally creates a copy of each node for each constraint cluster it belongs to. This may be inefficient for problems with many pairwise constraints, since each node must keep a copy of its parameter space corresponding to each constraint set it belongs to.

Other recent work has studied distributed optimization subject to subspace constraints by constructing local linear operators preserving a desired subspace [13], [14]. In this work, the relevant subspace is typically a space of bandlimited signals on a graph, and the local operator is designed to converge after iterated application to an orthogonal projection onto the desired subspace. Sheaf Laplacians are a straightforward source of such approximate projection operators, since for an appropriate choice of α, the matrix $I - \alpha L_{\mathcal{F}}$ satisfies the necessary requirements: as $n \to \infty$, $(I - \alpha L_{\mathcal{F}})^n$ becomes an orthogonal projection onto $H^0(G;\mathcal{F})$.

Since sheaf Laplacians are a class of local linear operators on networks with desirable properties, we expect that it will be possible to make use of them in other approaches to distributed optimization. Local linear constraints are naturally imposed in many problems, and sheaves are the natural way to handle these constraints in a distributed fashion.

V. Applications

In this section, we present a pair of straightforward applications of this distributed optimization framework inspired by the sheaves discussed in Section III-B.

A. Signal Recovery on Graphs

Suppose we have a collection of sensors located at nodes of a graph G, each of which observes the values of some function on the graph at each node in its neighborhood. That is, the sensor at vertex v observes the vector $\hat{x}_v = (f(u) + \epsilon_{uv})_{u \in \mathcal{N}(v)}$, where ϵ_{uv} is an independent error term for each pair (u, v). We wish to estimate f under the graph signal processing-inspired assumption that it is smooth: it has low variation over edges. This can be cast as a naturally distributed optimization problem over the sensing sheaf discussed in Section III-B.

We cover the vertex set of G by neighborhoods of vertices and construct the corresponding sheaf $\mathcal{F}$ of functions subordinate to this cover. Thus, the stalk $\mathcal{F}(v)$ is equal to $\mathbb{R}^{\mathcal{N}(v)}$, and the edge stalk $\mathcal{F}(e)$ for the edge between u and v is equal to $\mathbb{R}^{\{u,v\}}$. The restriction map $\mathcal{F}(v) \to \mathcal{F}(e)$ is the obvious projection map $\mathbb{R}^{\mathcal{N}(v)} \to \mathbb{R}^{\{u,v\}}$. The signal recovery problem is a tradeoff between smoothness and fidelity to the observations. A global formulation is as follows:

$$\min_{f:V(G)\to\mathbb{R}} \sum_{v\in V(G)} \sum_{u\in\mathcal{N}(v)} (f(u)-\hat{x}_v(u))^2 + a \sum_{u\sim v} (f(u)-f(v))^2.$$

To make this problem distributed, we optimize over 0-cochains of $\mathcal{F}$ instead of over functions on $V(G)$, imposing the constraint that these be sections of $\mathcal{F}$:

$$\min_{\mathbf{x}\in C^0(G;\mathcal{F})} \sum_{v\in V(G)} \left(\|x_v - \hat{x}_v\|^2 + \frac{a}{2} \sum_{u\sim v} (x_v(v) - x_v(u))^2 \right) \quad \text{s.t. } L_{\mathcal{F}}\mathbf{x} = 0. \tag{5}$$

The objective is now defined as a sum of functions, each acting on a single stalk of $\mathcal{F}$. Note that this particular choice of data fidelity and signal smoothness cost functions is not necessary; we could choose any smooth convex cost function.

The corresponding saddle point dynamics for this problem are

$$\begin{aligned}
\dot{x}_v(v) &= -2(x_v(v) - \hat{x}_v(v)) - a \sum_{u\sim v} (x_v(v) - x_v(u)) \\
&\quad -2 \sum_{u\sim v} (x_v(v) - x_u(v)) - \sum_{u\sim v} (z_v(v) - z_u(v)) \\
\dot{x}_v(u) &= -2(x_v(u) - \hat{x}_v(u)) - a(x_v(u) - x_v(v)) \\
&\quad -(x_v(u) - x_u(v)) - (z_v(u) - z_u(v)) \\
\dot{z}_v(v) &= \sum_{u\sim v} (x_v(v) - x_u(v)) \\
\dot{z}_v(u) &= x_v(u) - x_u(v),
\end{aligned}$$

where $u \in \mathcal{N}(v)$ and $u \neq v$.

B. Distributed Model Predictive Control

Take a coupled collection of discrete-time linear systems as discussed in Section III-B, and suppose that this coupling is not full—that is, not all inputs to a given subsystem are coupled to an output of another subsystem, so that the total coupled system can be seen as having a free input, distributed among the subsystems. A collection of local controllers is attached to each of these free inputs, and they wish to cooperate to steer the global system toward a particular state. We assume that the communication graph is the same as the coupling graph: a controller for system i may talk to the controllers for the systems to which system i is coupled.

We formulate this problem in the model predictive control setting, where at each time step the controllers wish to estimate a sequence of controls that optimally steer each system Σ_i to a particular state $\hat{x}_i$ within a finite time horizon k. We formalize the controllers' problem as follows:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \ & \sum_i f_i(x_i[k] - \hat{x}_i) + \sum_i \sum_{t=0}^{k-1} g_i(u_i[t]) \\
\text{s.t. } & x_i[0] = x_i^0 \\
& (\mathbf{x}, \mathbf{u}) \in H^0(G; \mathcal{M})
\end{aligned}$$

where f_i and g_i are convex cost functions on the terminal state and the control inputs. The extra local constraint coming from the initial state of the system requires a slight extension

of the optimization algorithm. An extra dual variable is added to the Lagrangian; since these constraints are local, the corresponding primal-dual dynamics are still local.

The saddle-point dynamics for this problem are more involved to describe than for the signal estimation problem. The augmented Lagrangian is

$$\mathcal{L}(\mathbf{x},\mathbf{u},\mathbf{z},\mu) = \sum_i f_i(x_i[k]-\hat{x}_i) + \sum_i \sum_{t=0}^{k-1} g_i(u_i[t]) + (\mathbf{x},\mathbf{u})^T L_{\mathcal{M}}(\mathbf{x},\mathbf{u}) + \sum_i \mu_i^T (x_i[0]-x_i^0) + \mathbf{z}^T L_{\mathcal{M}}(\mathbf{x},\mathbf{u})$$

This yields the saddle point dynamics

$$\begin{aligned}
\dot{x}_i[0] &= -\mu_i - 2(L_{\mathcal{M}}(x,u))_i^x[t] - (L_{\mathcal{M}}(z^x,z^u))_i^x[t] \\
\dot{x}_i[t] &= -2(L_{\mathcal{M}}(x,u))_i^x[t] - (L_{\mathcal{M}}(z^x,z^u))_i^x[t] \quad (0<t<k) \\
\dot{x}_i[k] &= -\nabla f_i(x_i[k]-\hat{x}_i) - 2(L_{\mathcal{M}}(x,u))_i^x[k] - (L_{\mathcal{M}}(z^x,z^u))_i^x[k] \\
\dot{u}_i[t] &= -\nabla g_i(u_i[t]) - 2(L_{\mathcal{M}}(x,u))_i^u[t] - (L_{\mathcal{M}}(z^x,z^u))_i^u[t] \\
\dot{z}_i^x[t] &= (L_{\mathcal{M}}(x,u))_i^x[t] \\
\dot{z}_i^u[t] &= (L_{\mathcal{M}}(x,u))_i^u[t] \\
\dot{\mu}_i &= x_i[0] - x_i^0
\end{aligned}$$

where

$$\begin{aligned}
&(L_{\mathcal{M}}(x,u))_i^x[t] \\
&\quad = \sum_{i\sim j} C_i^T Y_{ij}^T (Y_{ij}C_i x_i[t] + Y_{ij}D_i u_i[t] - U_{ji}u_j[t]) \\
&\qquad - A_i x_i[t-1] - B_i u_i[t-1] + (I + A_i^T A_i)x_i[t] \\
&\qquad\qquad + A_i^T B u_i[t] - A_i^T x_i[t+1]
\end{aligned}$$

and

$$\begin{aligned}
&(L_{\mathcal{M}}(x,u))_i^u[t] \\
&\quad = \sum_{i\sim j} D_i^T Y_{ij}^T (Y_{ij}C_i x_i[t] + Y_{ij}D_i u_i[t] - U_{ji}u_j[t]) \\
&\qquad + \sum_{i\sim j} U_{ij}^T (U_{ij}u_i[t] - Y_{ji}C_j x_j[t] - Y_{ji}D_j u_j[t]) \\
&\qquad\qquad + B_i^T A_i x_i[t] + B_i^T B_i u_i[t] - B_i^T x_i[t+1]
\end{aligned}$$

are the components of the sheaf Laplacian applied to (x,u). (For time indices less than 0 or greater than k we take the corresponding vectors to be zero.)

Although these dynamics are complicated to write explicitly, their derivation follows directly from the construction of the sheaf $\mathcal{M}$ and the definition of its Laplacian. No special insight is required to invent the optimization dynamics, only to deduce that the relevant constraints on admissible trajectories form the structure of a sheaf.

Note again that the specific structure of the cost functional is not crucial to the distributed implementation, only that it be defined locally on each stalk of $\mathcal{M}$.

VI. Simulations

To illustrate the behavior of the continuous-time dynamics, we generate a synthetic instance of the problem in Section V-A. $G = (V,E)$ is a random 4-regular graph with $N = 25$ nodes. The true function $f : V \to \mathbb{R}$ is generated by selecting first a standard multivariate normal function $g : V \to R$ and then smoothing it. The smoothing is done in the spectral domain of G, scaling the kth (graph Laplacian) frequency component of g by a Tikhonov-style scaling factor and then normalizing. To be precise, we have

$$g' = \sum_{i=1}^{N} \frac{1}{1+10\frac{\lambda_i}{\lambda_N}} \langle v_i, g\rangle v_i, \quad f = \frac{g'}{\|g'\|},$$

where (λ_i, v_i) is the eigenvalue-eigenvector pair corresponding to the ith largest eigenvalue of L_G. Figure 1 shows the graph G and the smooth signal f.

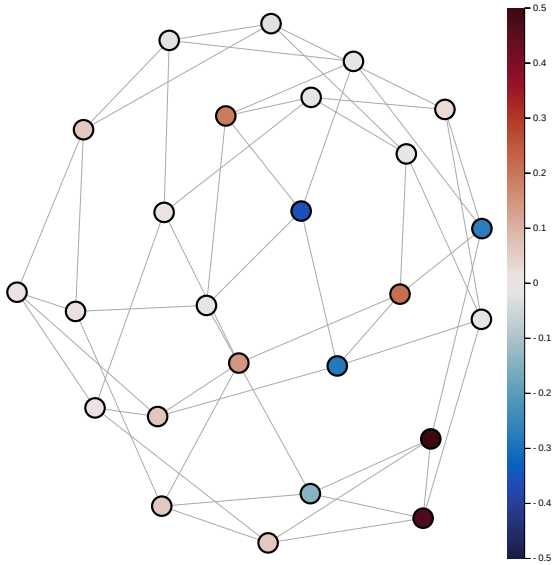

Fig. 1. The true function $f : V \to \mathbb{R}$.

We then create the sampled signal $\hat{\mathbf{x}}$ by adding mean zero Gaussian noise with variance $\sigma = 0.2$. This is a simple linear operation $\hat{\mathbf{x}} = Af + \sigma\mathbf{n}$, with A the linear map taking a function on the vertices of G to its corresponding element of $H^0(G;\mathcal{F})$, and $\mathbf{n}$ a vector of iid standard normal random variables. The Lagrangian saddle-point dynamics of the corresponding optimization problem are then implemented with a standard ODE solver. The initial conditions were $x_v(0) = \hat{x}_v$ and $\mathbf{z}(0) = 0$.

Since the non-distributed problem is simply an unconstrained quadratic program, we can calculate its solution exactly: $f_{\text{opt}} = (A^TA + aL_G)^{-1}A^T\hat{\mathbf{x}}$. We take the local estimate f_{loc} of the solution to consist of node i's estimate for f_i, ignoring the other data stored at node i. In Figure 2 we show the behavior of $\|f_{\text{opt}} - f_{\text{loc}}\|$ for a range of choices of a. We see that after a short period of quick improvement in the estimate of f_{opt}, the convergence levels off to a slower, but still exponential rate. In Figure 3 we compare f_{loc} to the true signal f, and see that for the purposes of estimating f, it is not necessary to run the dynamics for very long at all, since the estimation error $\|f_{\text{opt}} - f\|$ is larger than the distributed optimization error $\|f_{\text{opt}} - f_{\text{loc}}\|$ after only a short time.

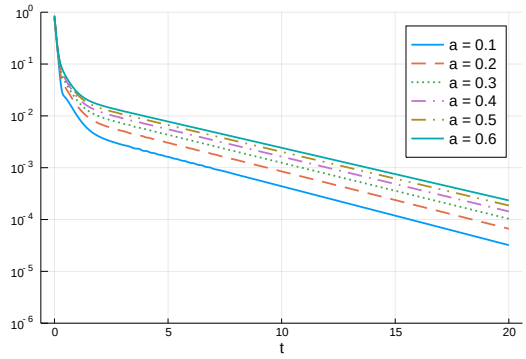

Fig. 2. Convergence of the local estimate f_{loc} to the optimum f_{opt}.

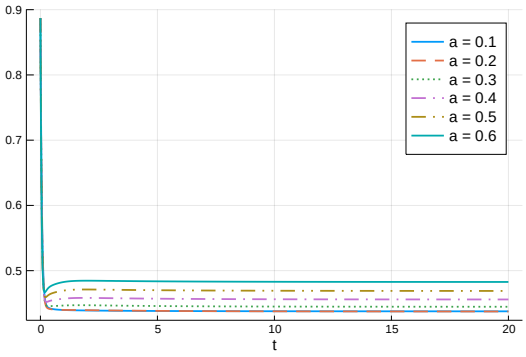

Fig. 3. Behavior of $\|f_{\text{loc}} - f\|$ under the distributed optimization dynamics.

VII. Conclusion

In this paper, we have proposed a method for distributed optimization with local linear constraints. These local linear constraints form the structure of a cellular sheaf on the graph, and restriction to the set of values satisfying these constraints may be seen as a *homological* constraint on the optimization domain. Cellular sheaf Laplacians give an immediate way to implement these constraints in distributed optimization dynamics, which then can be applied to various natural distributed optimization problems.

Cellular sheaves on graphs can only implement pairwise linear constraints, which is a significant limitation of this work compared with that in [11] and [12]. Higher-order linear constraints may be implemented with sheaves on simplicial complexes or cellular complexes, which can model higher-order relationships between agents. Laplacians of sheaves on higher-dimensional complexes generalize the Hodge Laplacians of simplicial complexes, which have previously been leveraged in distributed algorithms [15], [16]. Thus, it should be possible to use sheaves on complexes to alleviate the restriction to purely pairwise constraints. Another approach to higher-order constraints would be construction of a sheaf on a separate network associated with the constraint sets.

This paper has focused on using sheaf Laplacians to extend the work of Wang and Elia [8], but the framework of cellular sheaves and their Laplacians is quite general. We anticipate that sheaf Laplacians will prove useful in other distributed algorithms as well. The use of cellular sheaves and their Laplacians in systems design and analysis constitutes a homological approach to distributed systems, leveraging tools of algebraic topology and spectral graph theory to produce topologically inspired methods for concrete problems. The general framework of optimization described in this paper might be termed *homological programming*, and we expect that it be form a key tool for the topological study of distributed systems.

References

[1] J. Curry, "Sheaves, Cosheaves, and Applications," Ph.D. dissertation, University of Pennsylvania, 2014.

[2] J. Hansen and R. Ghrist, "Toward a Spectral Theory of Cellular Sheaves," *arXiv:1808.01513 [math]*, Aug. 2018. [Online]. Available: http://arxiv.org/abs/1808.01513

[3] 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, Sep. 1986.

[4] S. Boyd, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010. [Online]. Available: http://www.nowpublishers.com/article/Details/MAL-016

[5] M. Zhu and S. Martínez, "On distributed optimization under inequality and equality constraints via penalty primal-dual methods," in *Proceedings of the 2010 American Control Conference*, Jun. 2010, pp. 2434–2439.

[6] A. Nedic and A. Ozdaglar, "Distributed Subgradient Methods for Multi-Agent Optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4749425/

[7] A. Sayed, "Adaptation, Learning, and Optimization over Networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014. [Online]. Available: http://www.nowpublishers.com/articles/foundations-and-trends-in-machine-learning/MAL-051

[8] J. Wang and N. Elia, "Control approach to distributed optimization," in *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2010, pp. 557–561.

[9] ——, "A control perspective for centralized and distributed convex optimization," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec. 2011, pp. 3800–3805.

[10] A. Cherukuri, B. Gharesifard, and J. Cortés, "Saddle-Point Dynamics: Conditions for Asymptotic Stability of Saddle Points," *SIAM Journal on Control and Optimization*, vol. 55, no. 1, pp. 486–511, Jan. 2017. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/15M1026924

[11] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, "Diffusion LMS for Multitask Problems With Local Linear Equality Constraints," *IEEE Transactions on Signal Processing*, vol. 65, no. 19, pp. 4979–4993, Oct. 2017.

[12] F. Hua, R. Nassif, C. Richard, H. Wang, and J. Huang, "Penalty-based multitask distributed adaptation over networks with constraints," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct. 2017, pp. 908–912.

[13] P. D. Lorenzo, S. Barbarossa, and S. Sardellitti, "Distributed Signal Recovery Based on In-network Subspace Projections," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, May 2019, pp. 5242–5246. [Online]. Available: https://ieeexplore.ieee.org/document/8682719/

[14] R. Nassif, S. Vlaski, and A. H. Sayed, "Distributed Inference over Networks under Subspace Constraints," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 5232–5236.

[15] A. Muhammad and M. Egerstedt, "Control Using Higher Order Laplacians in Network Topologies," in *Proc. of 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto*, 2006, pp. 1024–1038.

[16] A. Tahbaz-Salehi and A. Jadbabaie, "Distributed Coverage Verification in Sensor Networks Without Location Information," *IEEE Transactions on Automatic Control*, vol. 55, no. 8, pp. 1837–1849, Aug. 2010.