Analysis and Design of First-Order Distributed Optimization Algorithms over Time-Varying Graphs

Akhil Sundararajan, Bryan Van Scoy, and Laurent Lessard

***Abstract*— This work concerns the analysis and design of distributed first-order optimization algorithms over time-varying graphs. The goal of such algorithms is to optimize a global function that is the average of local functions using only local computations and communications. Several different algorithms have been proposed that achieve linear convergence to the global optimum when the local functions are strongly convex. We provide a unified analysis that yields the worst-case linear convergence rate as a function of the condition number of the local functions, the spectral gap of the graph, and the parameters of the algorithm. The framework requires solving a small semidefinite program whose size is fixed; it does not depend on the number of local functions or the dimension of their domain. The result is a computationally efficient method for distributed algorithm analysis that enables the rapid comparison, selection, and tuning of algorithms. Finally, we propose a new algorithm, which we call SVL, that is easily implementable and achieves a faster worst-case convergence rate than all other known algorithms.**

I. Introduction

In distributed optimization, a network of agents, such as computing nodes, robots, or mobile sensors, work collaboratively to optimize a global objective. Specifically, each agent $i \in \{1,\dots,n\}$ has access to a local function f_i and must minimize the average of all agents' local functions

$$\min_{x\in\mathbb{R}^d} f(x), \quad \text{where } f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{1}$$

by querying its local gradient ∇f_i, exchanging information with neighboring agents, and performing local computations.

This work aims to study the reliability of distributed optimization algorithms in the presence of a *time-varying* communication graph. Such a scenario could occur if communication links fail due to interference, mobile agents move out of range, or an adversary is jamming communications.

A. Sundararajan is with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI 53706 USA (e-mail: asundararaja@wisc.edu).

B. Van Scoy is with the Wisconsin Institute for Discovery, Madison, WI 53715 USA (e-mail: vanscoy@wisc.edu).

L. Lessard is with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI 53706 USA, and the Wisconsin Institute for Discovery, Madison, WI 53715 USA (e-mail: laurent.lessard@wisc.edu).

Digital Object Identifier 10.1109/TCNS.2020.2988009

Distributed optimization is relevant in many application areas. For example, in large-scale machine learning [7], [9], n could represent the number of computing units available for training a large data set. Each f_i then denotes the loss function corresponding to the training examples assigned to unit i. Another example is sensor networks [20], where each sensor may have a limited power budget, communication bandwidth, or sensing capability. The goal is to aggregate all local data without having a single point of failure. Other applications include distributed spectrum sensing [2] and resource allocation across geographic regions [21].

Distributed optimization generalizes both average consensus and centralized optimization, as we now explain.

a) Consensus: If each agent uses the initial value x_i^0 and local objective $f_i(x) = \|x - x_i^0\|^2$, distributed optimization reduces to *average consensus* [27], [29]. The unique optimizer of (1) is then the average of all initial states: $x^\star = \frac{1}{n}\sum_{i=1}^n x_i^0$. Using a *gossip* update of the form $x_i^{k+1} = \sum_{i=1}^n W_{ij} x_j^k$ where W is carefully chosen, such methods converge exponentially: $\|x_i^k - x^\star\| \le \rho^k$ with $\rho \in (0,1)$ that depends on W [30]. This is called a *linear rate* in the optimization community.

b) Optimization: If $n = 1$ or if all f_i are identical, we recover the standard centralized optimization setup. Linear convergence can be guaranteed in certain cases. For example, the gradient descent method $x_i^{k+1} = x_i^k - \alpha\nabla f_i(x_i^k)$ achieves linear convergence if f_i is continuously differentiable, smooth, and strongly convex (formally stated in Assumption 1) [16].

A linear convergence rate for the general case was first achieved by the *exact first-order algorithm* (EXTRA) [23]. This algorithm requires storing the previous state in memory:

$$x_i^1 = \sum_{j=1}^n W_{ij}\, x_j^0 - \alpha\,\nabla f_i(x_i^0), \qquad x_i^0 \text{ arbitrary}, \tag{2a}$$

$$x_i^{k+2} = x_i^{k+1} + \sum_{j=1}^n W_{ij}\, x_j^{k+1} - \sum_{j=1}^n \widetilde{W}_{ij}\, x_j^k - \alpha\left(\nabla f_i(x_i^{k+1}) - \nabla f_i(x_i^k)\right) \tag{2b}$$

where W and $\widetilde{W}$ are gossip matrices that satisfy certain technical conditions and α is sufficiently small. Several additional linear-rate algorithms have since been proposed, including: AugDGM [34], DIGing [15], [19], Exact Diffusion [35], [36], NIDS [13], and a unified method [8]. Each of these methods

have updates similar to (2) in that they require agents to store previous iterates or gradients.

Although linear convergence rates were obtained for the algorithms above, each algorithm differs in the nature and strength of its convergence analysis guarantees. For example, some works show (non-constructively) the existence of a linear rate [32] whereas others provide specific tuning recommendations with associated analytic rate bounds (which may be conservative) [13], [23]. Numerical simulations are also frequently used [31], but can be misleading because algorithm performance depends on the graph topology, choice of functions, algorithm initialization, and algorithm tuning.

The present work makes an effort to systematize the analysis and design of distributed optimization algorithms. We now summarize our main contributions.

Analysis framework. We present a universal analysis framework that provides an upper bound on the worst-case linear convergence rate ρ of a wide range of distributed algorithms as a function of the parameters κ (local function conditioning) and σ (network connectedness). Our main result, Theorem 10, is a semidefinite program (SDP) parameterized by (κ, σ) whose solution yields an upper bound on ρ. The SDP has a small fixed size that does not depend on the number of agents n or the dimension of the function domains and is efficiently solvable. Our SDP yields robust performance guarantees when the graph is allowed to vary (even adversarially) at each iteration. Fig. 2 compares the worst-case linear rate ρ for 8 different algorithms.

Algorithm design. We present a new distributed algorithm, which we name SVL (the authors' initials). SVL is derived by optimizing the SDP from our analysis framework and provides the fastest known convergence rate to date for this time-varying graph setting. The rate depends explicitly on κ and σ, so no tuning is required if these parameters are known or estimated in advance. When the graph is well-connected, SVL recovers the performance of gradient descent, which is optimal in this time-varying graph setting.

Worst-case examples. Although our analysis technique only provides *upper bounds* on the worst-case convergence rate for distributed algorithms, we outline a computationally tractable optimization procedure that finds numerically matching lower bounds by constructing worst-case trajectories, suggesting the bounds found via our analysis technique are tight.

Remark 1 (Accelerated rates): Distributed algorithms that achieve *accelerated* [18], [31], [33] or *optimal* [22] linear rates have also been proposed. It turns out such methods are not guaranteed to achieve acceleration when the graph is time-varying. We discuss this phenomenon in Section II-E, where we derive lower bounds for the time-varying setting.

The paper is organized as follows. We describe notation and assumptions in Section II. We state and prove our main result for certifying worst-case rate bounds in Section III. We present our SVL algorithm and discuss interpretations in Section IV. Finally, we demonstrate the tightness of our bounds by generating worst-case trajectories in Section V.

II. Preliminaries

A. Notation

Let I_n be the identity matrix in $\mathbb{R}^{n\times n}$. The symbol 1_n denotes the column vector of all ones in $\mathbb{R}^n$. $\Pi := \frac{1}{n}1_n 1_n^\mathsf{T}$ is the projection matrix onto 1_n. We will sometimes omit subscripts when dimensions are clear from context. Unless otherwise indicated, Greek letters denote scalar parameters, lower-case letters denote column vectors, and upper-case letters denote matrices. Exceptions include the scalars m and L, which we use in Assumption 1 to conform with convention. The symbol $\otimes$ denotes the Kronecker matrix product. $\|x\|$ denotes the standard Euclidean norm of a vector x, and $\|A\| := \sup_{x\neq 0}\|Ax\|/\|x\|$ is the spectral norm of a matrix A. Unless otherwise indicated, subscripts refer to individual agents while superscripts refer to iteration count. For brevity, we write the symmetric quadratic form $x^\mathsf{T}Qx$ as $\left[\star\right]^\mathsf{T}Qx$.

Define the graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} := \{1, \ldots, n\}$ is the set of agents and $\mathcal{E}$ is the set of pairs of agents (i, j) that are connected. $\mathcal{L} \in \mathbb{R}^{n\times n}$ is a *Laplacian matrix* associated with $\mathcal{G}$ if $\mathcal{L}1_n = 0$ and $\mathcal{L}_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. The *spectral gap* of $\mathcal{L}$ is defined as the second-smallest eigenvalue magnitude of $\mathcal{L}$. Since we consider time-varying graphs, we let $\mathcal{L}^k$ denote a Laplacian matrix associated with $\mathcal{G}^k$.

We denote a symbol on agent i at iteration k by x_i^k along with its associated fixed point $x_i^\star$. For all such symbols, we denote their aggregation over all agents as

$$x^k := \begin{bmatrix} x_1^k \\ \vdots \\ x_n^k \end{bmatrix} \quad \text{and} \quad x^\star = \begin{bmatrix} x_1^\star \\ \vdots \\ x_n^\star \end{bmatrix}.$$

We denote the associated local and global error coordinates as $\tilde{x}_i^k := x_i^k - x_i^\star$ and $\tilde{x}^k := x^k - x^\star$, respectively.

B. Function and Graph Assumptions

We assume that the local function gradients satisfy the following *sector bound*.

Assumption 1: Given $0 < m \leq L$, the local objective functions f_i are continuously differentiable and each satisfy

$$\begin{aligned}\big(\nabla f_i(y) - \nabla f_i(y_{\text{opt}}) - m\,(y - y_{\text{opt}})\big)^\mathsf{T} \\ \times \big(\nabla f_i(y) - \nabla f_i(y_{\text{opt}}) - L\,(y - y_{\text{opt}})\big) \leq 0\end{aligned}$$

for all $y \in \mathbb{R}^d$, where y_{opt} satisfies $\sum_{i=1}^n \nabla f_i(y_{\text{opt}}) = 0$.

Remark 2: One way to satisfy Assumption 1 is if the local functions f_i are m-strongly convex with L-Lipschitz continuous gradients, though in general Assumption 1 is weaker.

We define the condition ratio as $\kappa := L/m$. This quantity captures the amount of variation in the curvature of the objective function. If f_i is twice differentiable, κ is an upper bound on the condition number of the Hessian $\nabla^2 f_i$. In general, as $\kappa \to \infty$, the functions become poorly conditioned and more difficult to optimize using first-order methods.

The graph associated with the network of agents can change at each step of the algorithm, so we assume the following about the sequence of graph Laplacian matrices $\{\mathcal{L}^k\}$.

Assumption 2: The following properties hold at each step of the algorithm.

1) The graph is connected: there always exists a path between any two nodes in $\mathcal{G}^k$. This implies that the zero eigenvalue of $\mathcal{L}^k$ has a multiplicity of one for all k.
2) The graph is balanced: every node has equal in-degree and out-degree. This means that $1_n^\mathsf{T}\mathcal{L}^k = 0$ for all k.
3) The spectral gap of the time-varying graph is uniformly bounded. In particular, we assume there exists $\sigma \in [0,1)$ such that $\|I - \Pi - \mathcal{L}^k\| \leq \sigma$ for all k. Since the spectral radius of a matrix is always upper-bounded by its spectral norm, this implies that σ is a uniform bound on the spectral gap of each Laplacian matrix in $\{\mathcal{L}^k\}$.

Remark 3: The assumption that $\mathcal{G}^k$ must be connected for all k is a strong assumption. Works that consider directed or time varying graphs typically make weaker assumptions, such as a *joint spectrum property* or B-*connectedness* [15]. Nevertheless, our setting (which is equivalent to B-connectedness with $B = 1$) is still weaker than assuming a constant graph. Indeed, NIDS [13] converges for any σ when the graph is constant, but in Section V-B, we construct a sequence of graphs that drives NIDS to instability.

C. Algorithm Form

In this paper, we consider the broad class of distributed optimization algorithms that satisfy the algebraic equations

$$\begin{bmatrix} x_i^{k+1} \\ y_i^k \\ z_i^k \end{bmatrix} = \begin{bmatrix} A & B_u & B_v \\ C_y & D_{yu} & D_{yv} \\ C_z & D_{zu} & D_{zv} \end{bmatrix} \begin{bmatrix} x_i^k \\ u_i^k \\ v_i^k \end{bmatrix}, \tag{3a}$$

$$u_i^k = \nabla f_i(y_i^k), \qquad v_i^k = \sum_{j=1}^n \mathcal{L}_{ij}^k z_j^k, \tag{3b}$$

$$\sum_{j=1}^n \left(F_x x_j^k + F_u u_j^k\right) = 0. \tag{3c}$$

Equation (3a) describes how agent i's state x_i^k evolves with iteration k. The local gradient ∇f_i is evaluated at y_i^k and the quantity z_i^k is transmitted to neighboring agents in (3b). Finally, we allow for linear state-input invariants to be enforced in (3c). Such invariants typically arise from requiring a particular initialization for the algorithm.

The matrices A, D_{yu}, and D_{zv} are square, and the other matrices have compatible dimensions. The dimension of A is the number of local states on each agent, the dimension of D_{yu} is one, and the dimension of D_{zv} is the number of variables that each agent transmits with neighbors at each iteration.

Remark 4 (Dimension reduction): To simplify notation, we assume the objective function is one-dimensional ($d = 1$). We can recover the general d case by replacing each scalar symbol with a $1 \times d$ row vector (e.g., $u_i^k \in \mathbb{R}^{1\times d}$) and interpreting each local gradient ∇f_i as a map from $\mathbb{R}^{1\times d}$ to $\mathbb{R}^{1\times d}$.

Remark 5 (Implementation): Not all instances of (3) are efficiently implementable. For example, if $D_{yu} \neq 0$, then y_i^k depends on u_i^k, which then depends on y_i^k. Such circular dependencies arise naturally in proximal algorithms, where an inner optimization problem must be solved at each iteration. For instance, given a convex differentiable f and parameter $\lambda > 0$, the proximal algorithm

$$x^{k+1} = \mathbf{prox}_{\lambda f}(x^k) := \arg\min_x \left(\lambda f(x) + \tfrac{1}{2}\|x - x^k\|^2\right)$$

satisfies the optimality condition $\lambda\nabla f(x^{k+1}) + x^{k+1} - x^k = 0$ and can therefore be expressed in the form of (3) as follows:

$$x^{k+1} = x^k - \lambda u^k, \quad y^k = x^k - \lambda u^k, \quad u^k = \nabla f(y^k).$$

In the forthcoming analysis, we treat implementability and analysis separately. That is, we derive convergence rate bounds for general algorithms of the form (3), regardless of whether they can be efficiently implemented. However, we note that a sufficient condition for avoiding circular dependencies is if the feedthrough term satisfies

$$\begin{bmatrix} D_{yu} & D_{yv} \\ D_{zu} & D_{zv} \end{bmatrix} = \begin{bmatrix} 0 & D_{yv} \\ 0 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 0 & 0 \\ D_{zu} & 0 \end{bmatrix}. \tag{4}$$

Putting a distributed optimization algorithm into the form of (3) is a straightforward algebraic exercise, which we now demonstrate for two recently proposed algorithms. These algorithms are parameterized by a stepsize α and a gossip matrix W. To relate the gossip matrix to the Laplacian matrix, we set $W = I - \mu\mathcal{L}$ for some scalar $\mu \neq 0$. This provides an additional tuning parameter, and is akin to the method of successive overrelaxation used in the numerical solutions of linear systems of equations [17].

a) EXTRA: The EXTRA algorithm (2) has a state that depends on two previous timesteps. Using the authors' recommendation of $\widetilde{W} = \frac{1}{2}(I + W)$ together with $W = I - \mu\mathcal{L}^k$, the equations become

$$\begin{aligned} x^1 &= x^0 - \alpha\nabla f(x^0) - \mu\mathcal{L}^k x^0, \\ x^{k+2} &= 2x^{k+1} - x^k - \alpha\left(\nabla f(x^{k+1}) - \nabla f(x^k)\right) \\ &\qquad - \mu\mathcal{L}^k\left(x^{k+1} - \tfrac{1}{2}x^k\right). \end{aligned}$$

Define the state $(x^{k+1}, x^k, \nabla f(x^k))$. The outputs are now functions of the state: $y^k := x^{k+1}$ and $z^k := x^{k+1} - \frac{1}{2}x^k$. Finally, summing across agents (left-multiplying by 1^T) and using $1^\mathsf{T}\mathcal{L}^k = 0$, we find that $1^\mathsf{T}\left(x^{k+1} - x^k + \alpha\nabla f(x^k)\right)$ is independent of k, and identically zero thanks to how x^1 is initialized. The parameters that characterize EXTRA are shown below and in Table I.

$$\left[\begin{array}{c|cc} A & B_u & B_v \\ \hline C_y & D_{yu} & D_{yv} \\ C_z & D_{zu} & D_{zv} \\ \hline F_x & F_u & \end{array}\right] = \left[\begin{array}{ccc|cc} 2 & -1 & \alpha & -\alpha & -\mu \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ \hline 1 & -\frac{1}{2} & 0 & 0 & 0 \\ \hline 1 & -1 & \alpha & 0 & \end{array}\right].$$

b) DIGing: The DIGing algorithm [15], [19], is an example of a *gradient tracking* algorithm. It begins with an arbitrary x^0 and has two update equations:

$$\begin{aligned} s^0 &= \nabla f(x^0), \\ x^{k+1} &= Wx^k - \alpha s^k, \\ s^{k+1} &= \widetilde{W}s^k + \nabla f(x^{k+1}) - \nabla f(x^k). \end{aligned}$$

Using the authors' recommendation of $\widetilde{W} = W$, defining $W = I - \mu\mathcal{L}^k$ as before, and defining the state as $(x^k, s^k, \nabla f(x^k))$, we find that the output is $y^k := x^{k+1}$, two quantities must be communicated between agents, $z^k := (x^k, s^k)$, and the invariant is $1^\mathsf{T}(s^k - \nabla f(x^k)) = 0$. The parameters that characterize DIGing are shown in Table I.

A similar derivation can be applied to a variety of algorithms. Table I summarizes the parameterizations for 8 recently proposed algorithms.

D. Existence of a Fixed Point

Not all instances of algorithm (3) solve the distributed optimization problem (1). For an algorithm to be valid, (i) there must exist a fixed point corresponding to the optimal solution, and (ii) the iterates must converge to the fixed point. We address convergence to a fixed point in our main result of Section III. In this section, however, we provide simple conditions for verifying the existence of such a fixed point.

A distributed algorithm of the form (3) has a fixed point $(x^\star, y^\star, z^\star, u^\star, v^\star)$ corresponding to the optimal solution of (1) for all functions satisfying Assumption 1 and all graphs satisfying Assumption 2 if the following conditions hold.

- **Consensus and Optimality:** All agents must achieve consensus on the point at which the gradient is evaluated, and the point must be a stationary (first-order optimal) point of f. This means that the fixed point must satisfy $y_1^\star = \ldots = y_n^\star$ and $u_1^\star + \cdots + u_n^\star = 0$, or in vector form,
$$(I - \Pi)\, y^\star = 0 \quad \text{and} \quad 1^\mathsf{T} u^\star = 0. \tag{5a}$$
- **Robustness to Graph:** The fixed point must not depend on the sequence of graphs $\{\mathcal{L}^k\}$, so $z_1^\star = \ldots = z_n^\star$ and $v_1^\star = \cdots = v_n^\star = 0$, or in vector form,
$$(I - \Pi)\, z^\star = 0 \quad \text{and} \quad v^\star = 0. \tag{5b}$$
- **Robustness to Functions:** The fixed point must satisfy $y_1^\star = \ldots = y_n^\star = y_{\text{opt}}$ and $u_i^\star = \nabla f_i(y_{\text{opt}})$, where y_{opt} is the optimizer of (1). For these to hold for any objective function f, we need
$$1^\mathsf{T} y^\star \text{ and } (I - \Pi)\, u^\star \text{ unconstrained.} \tag{5c}$$

The following proposition characterizes algorithms with such a fixed point, which we prove in Appendix A.

Proposition 6 (Existence of fixed point): An algorithm of the form (3) has a fixed point $(x^\star, y^\star, z^\star, u^\star, v^\star)$ that satisfies the conditions in (5) if and only if
$$\text{null}(A - I) \cap \text{row}(C_y) \cap \text{null}(F_x) \neq \{0\} \tag{6a}$$
$$\text{and} \quad \begin{bmatrix} B_u \\ D_{yu} \\ D_{zu} \end{bmatrix} \in \text{col}\left(\begin{bmatrix} A - I \\ C_y \\ C_z \end{bmatrix} \right). \tag{6b}$$

Here, "null", "col", and "row" denote the nullspace, column space, and row space, respectively. Both EXTRA and DIGing as derived above satisfy the conditions in (6) and therefore have a fixed point corresponding to the optimal solution of (1).

Remark 7: Proposition 6 guarantees that any instance of algorithm (3) satisfying (5) has a desirable fixed point in the presence of a time-varying graph; all agents agree on a common stationary point of (1). However, Proposition 6 does not ensure that the algorithm necessarily converges to this fixed point, nor does it characterize the rate of convergence. These questions will be explored in Section III.

E. Lower Bounds on Worst-Case Convergence Rates

We now construct simple lower bounds on the worst-case asymptotic convergence rate of the iterates for any valid algorithm of the form (3). We do so by separately considering the two specific instances discussed in Section I.

a) Consensus: Consider the scalar local quadratic functions $f_i(y) = \frac{L}{2}(y - r_i)^2$. Then Assumption 1 holds with $m = L$ and $y_{\text{opt}} = \frac{1}{n}\sum_{i=1}^n r_i$.

b) Optimization: Consider the case $n = 1$. For the graph to satisfy Assumption 2, the Laplacian matrix must be $\mathcal{L}^k = 0$, which has spectral gap $\sigma = 0$.

In both cases above, the algorithm reduces to a linear system in feedback with sector-bounded nonlinearity: in the sector $(1 - \sigma, 1 + \sigma)$ for consensus and (m, L) for optimization. Further, the linear part of the system is strictly proper (since the algorithm is implementable) and must contain an integrator (due to the fixed-point conditions). Then, using the lower bound for such systems in [12], we obtain the following.

Proposition 8: For any $\rho < \max\left\{\frac{\kappa-1}{\kappa+1}, \sigma\right\}$, there does *not* exist an algorithm of the form (3) that satisfies the implementability conditions (4) and fixed-point conditions (6) such that, for all objective functions and Laplacian matrices satisfying Assumptions 1 and 2, there exists a constant $c > 0$ such that the bound $\|x_i^k - y_{\text{opt}}\| \leq c\,\rho^k$ holds for all agents $i \in \{1, \ldots, n\}$ and all iterations $k \geq 0$.

Remark 9 (Accelerated rates): These lower bounds, which are achieved by ordinary gradient descent, imply that accelerated algorithms such as the recently proposed SSDA [22] or distributed versions of heavy-ball [33] or Nesterov acceleration [18], [31] do not in fact achieve accelerated rates in the worst case when the local function gradients are sector bounded and the graph is time-varying as in Assumptions 1 and 2, respectively.

III. Main Result

Our main theorem, Theorem 10, consists of a small convex semidefinite program (SDP) whose feasibility guarantees the linear convergence of a distributed algorithm in the form of (3). The algorithm parameters, problem data (κ, σ), and candidate linear rate ρ all appear as parameters in the SDP. Furthermore, the SDP has a fixed size that does not depend on n (the number of agents) or d (the dimension of the domain of f) and can thus be efficiently solved using a variety of established solvers.

Theorem 10 (Analysis result): Consider the distributed optimization problem (1) solved using algorithm (3). Suppose Assumptions 1 and 2 hold and further assume the algorithm satisfies the fixed point conditions (6). Define the matrices
$$M_0 := \begin{bmatrix} -2mL & L + m \\ L + m & -2 \end{bmatrix} \quad \text{and} \quad M_1 := \begin{bmatrix} \sigma^2 - 1 & 1 \\ 1 & -1 \end{bmatrix}.$$
Let Ψ be a matrix whose columns form a basis for the nullspace of $\begin{bmatrix} F_x & F_u \end{bmatrix}$. If there exist $P \succ 0$, $Q \succ 0$, and

TABLE I

ALGORITHM PARAMETERS IN THE FORM OF (3) FOR A VARIETY OF DIFFERENT DISTRIBUTED OPTIMIZATION ALGORITHMS. ALGORITHMS CAN BE TUNED BY CHOOSING STEPSIZE AND OVERRELAXATION PARAMETERS $\boldsymbol{\alpha}$ AND $\boldsymbol{\mu}$, RESPECTIVELY. ALGORITHMS ARE ORGANIZED BASED ON HOW MANY INTERNAL STATES THEY HAVE (COLUMNS) AND HOW MANY VARIABLES MUST BE COMMUNICATED IN EACH ITERATION (BLOCK ROWS).

	Algorithms with 2 states		Algorithms with 3 states	
1 communicated variable	**SVL template (present work)** See Section IV for derivation of $(\alpha,\beta,\gamma,\delta)$	$\left[\begin{array}{cc:c:c} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \\ \hdashline 1 & 0 & 0 & -\delta \\ \hdashline 1 & 0 & 0 & 0 \\ \hdashline 0 & 1 & 0 & \end{array}\right]$	EXTRA [23]	$\left[\begin{array}{cc:c:c:c} 2 & -1 & \alpha & -\alpha & -\mu \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & -\frac{1}{2} & 0 & 0 & 0 \\ \hdashline 1 & -1 & \alpha & 0 & \end{array}\right]$
	Exact Diffusion (ExDIFF) [35], [36]	$\left[\begin{array}{cc:c:c} 2 & -1 & -\alpha & -\mu \\ 1 & 0 & -\alpha & -\frac{1}{2}\mu \\ \hdashline 1 & 0 & -\frac{1}{2}\mu & 0 \\ \hdashline 1 & 0 & 0 & 0 \\ \hdashline 1 & -1 & 0 & \end{array}\right]$	NIDS [13]	$\left[\begin{array}{cc:c:c:c} 2 & -1 & \alpha & -\alpha & -\mu \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & -\frac{1}{2} & \frac{\alpha}{2} & -\frac{\alpha}{2} & 0 \\ \hdashline 1 & -1 & \alpha & 0 & \end{array}\right]$
2 communicated variables	Unified DIGing (uDIG) [8]	$\left[\begin{array}{cc:c:c:c} 1 & -\alpha & -\alpha & -\mu & 0 \\ 0 & 1 & 0 & 0 & -\mu \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline -\frac{L+m}{2} & 1 & 1 & 0 & 0 \\ \hdashline 0 & 1 & 0 & & \end{array}\right]$	DIGing [15], [19]	$\left[\begin{array}{cc:c:c:c:c} 1 & -\alpha & 0 & 0 & -\mu & 0 \\ 0 & 1 & -1 & 1 & 0 & -\mu \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 1 & -\alpha & 0 & 0 & -\mu & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 1 & -1 & 0 & & \end{array}\right]$
	Unified EXTRA (uEXTRA) [8]	$\left[\begin{array}{cc:c:c:c} 1 & -\alpha & -\alpha & -\mu & 0 \\ 0 & 1 & 0 & 0 & -\mu \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline 1 & 0 & 0 & 0 & 0 \\ \hdashline -L & 1 & 1 & L\mu & 0 \\ \hdashline 0 & 1 & 0 & & \end{array}\right]$	AugDGM [34]	$\left[\begin{array}{cc:c:c:c:c} 1 & -\alpha & 0 & 0 & -\mu & \alpha\mu \\ 0 & 1 & -1 & 1 & 0 & -\mu \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hdashline 1 & -\alpha & 0 & 0 & -\mu & \alpha\mu \\ \hdashline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 1 & -1 & 0 & & \end{array}\right]$

$R \succeq 0$ of appropriate sizes such that

$$\Psi^{\mathsf{T}} \left[\star\right]^{\mathsf{T}} \left[\begin{array}{cc:c} P & 0 & 0 \\ 0 & -\rho^2 P & 0 \\ \hdashline 0 & 0 & M_0 \end{array}\right] \left[\begin{array}{cc} A & B_u \\ I & 0 \\ \hdashline C_y & D_{yu} \\ 0 & I \end{array}\right] \Psi \preceq 0 \quad \text{(7a)}$$

$$\left[\star\right]^{\mathsf{T}} \left[\begin{array}{cc:cc} Q & 0 & 0 & 0 \\ 0 & -\rho^2 Q & 0 & 0 \\ \hdashline 0 & 0 & M_0 & 0 \\ 0 & 0 & 0 & M_1 \otimes R \end{array}\right] \left[\begin{array}{ccc} A & B_u & B_v \\ I & 0 & 0 \\ \hdashline C_y & D_{yu} & D_{yv} \\ 0 & I & 0 \\ \hdashline C_z & D_{zu} & D_{zv} \\ 0 & 0 & I \end{array}\right] \preceq 0 \quad \text{(7b)}$$

then there exists a constant[1] $c > 0$ independent of i and k such that for all agents $i \in \{1, \dots, n\}$ and all iterations $k \geq 0$,

$$\|x_i^k - x_i^\star\| \leq c\, \rho^k \quad (8)$$

for some fixed point $(x_i^\star, y_i^\star, z_i^\star, u_i^\star, v_i^\star)$ that satisfies (5).

For fixed algorithm parameters $A, B_u, B_v, C_y, C_z, D_{yu}, D_{yv}, D_{zu}, D_{zv}, F_x, F_u$, function parameters m and L, graph parameter σ, and candidate rate ρ, the SDP (7) is a linear matrix inequality (LMI) in the variables (P, Q, R), and therefore convex. Indeed, (7a) and (7b) are decoupled and their feasibility may be checked separately. To find the best (smallest) upper bound, we observe that feasibility of (7) for some ρ_0 implies feasibility for all $\rho \geq \rho_0$. A bisection search on ρ is then guaranteed to find the minimal ρ, even though (7) is not jointly convex in (P, Q, R, ρ). While our result is only a sufficient condition for convergence, we provide empirical evidence in Section V-B that suggests that it is in fact tight.

Remark 11: Our main theorem provides conditions under which the *state* converges to a fixed point linearly with rate ρ. However, when the algorithm also satisfies the conditions in (4) for being efficiently implementable, then under the conditions of Theorem 10, there exist constants c_u, c_v, c_y, and c_z such that for all agents i and all iterations k,

$$\|u_i^k - u_i^\star\| \leq c_u\, \rho^k, \qquad \|y_i^k - y_i^\star\| \leq c_y\, \rho^k,$$
$$\|v_i^k - v_i^\star\| \leq c_v\, \rho^k, \qquad \|z_i^k - z_i^\star\| \leq c_z\, \rho^k,$$

for some fixed point $(x_i^\star, y_i^\star, z_i^\star, u_i^\star, v_i^\star)$ that satisfies (5). In particular, the output sequence y_i^k of each agent converges to the optimizer y_{opt} of (1) linearly with rate ρ.

The core idea behind Theorem 10 is to posit a quadratic Lyapunov candidate of the form

$$V^k := (x^k - x^\star)^{\mathsf{T}} \big(\Pi \otimes P + (I - \Pi) \otimes Q\big)(x^k - x^\star) \quad (9)$$

for some appropriate choice of $P \succ 0$ and $Q \succ 0$. Feasibility of (7) can be shown to imply $V^{k+1} \leq \rho^2 V^k$, which ensures linear convergence of the distributed optimization algorithm when $\rho < 1$. A preliminary (and less concise) version of Theorem 10 appeared in [25]. The proof of Theorem 10 is given in Appendix B.

[1]The constant c does not depend on the particular local functions f_i and sequence $\{\mathcal{L}^k\}$, but does depend on $x_i^0 - x_i^\star$. We provide an explicit formula for c in the proof in Appendix B.

IV. Algorithm Design

We now use Theorem 10 to design a distributed optimization algorithm, which we name SVL. Our guiding principle is to seek the fastest possible rate bound guarantee while keeping the algorithm as simple as possible. Therefore, we seek an algorithm with two states that only requires one state to be communicated at every timestep. Inspired by our previous work in which we developed a canonical form for distributed algorithms over time-invariant graphs [26], we restrict our search to algorithms of the form (3) with

$$\left[\begin{array}{c|c|c} A & B_u & B_v \\ \hline C_y & D_{yu} & D_{yv} \\ \hline C_z & D_{zu} & D_{zv} \\ \hline F_x & F_u & \end{array}\right] = \left[\begin{array}{cc|cc} 1 & \beta & -\alpha & -\gamma \\ 0 & 1 & 0 & -1 \\ \hline 1 & 0 & 0 & -\delta \\ \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & \end{array}\right]. \tag{10}$$

As long as $\beta \neq 0$, this algorithm satisfies the fixed point conditions of Proposition 6. Moreover, the update equations satisfy (4) and therefore do not contain circular dependencies, so we can implement the algorithm in a straightforward fashion as in Algorithm 1. To motivate the structure of our algorithm, we show how it corresponds to an inexact version of the alternating direction method of multipliers (ADMM), as well as how it reduces to well-known consensus and optimization algorithms in special cases. But first, we show how to use the SDP (7) to choose the algorithm parameters.

Algorithm 1 (template for the SVL algorithm)

Initialization: Let $\mathcal{L}^k \in \mathbb{R}^{n\times n}$ be a Laplacian matrix. Agents $i \in \{1,\dots,n\}$ choose initial local state $x_i^0 \in \mathbb{R}^d$ arbitrarily and $w_i^0 \in \mathbb{R}^d$ such that $\sum_{i=1}^n w_i^0 = 0$ (e.g. $w_i^0 = 0$).

for iteration $k = 0, 1, 2, \dots$ **do**

  for agent $i \in \{1,\dots,n\}$ **do**

    Local communication

$$v_i^k = \textstyle\sum_{j=1}^n \mathcal{L}_{ij}^k x_j^k \tag{C.1}$$

    Local gradient computation

$$y_i^k = x_i^k - \delta\, v_i^k \tag{C.2}$$

$$u_i^k = \nabla f_i(y_i^k) \tag{C.3}$$

    Local state update

$$x_i^{k+1} = x_i^k + \beta\, w_i^k - \alpha\, u_i^k - \gamma\, v_i^k \tag{C.4}$$

$$w_i^{k+1} = w_i^k - v_i^k \tag{C.5}$$

  end for

end for

A. Choosing the Algorithm Parameters

The problem of minimizing the worst-case convergence rate ρ over the algorithm parameters $(\alpha,\beta,\gamma,\delta)$ and SDP solution (P,Q,R) subject to the SDP being feasible is difficult due to the nonlinear matrix inequalities (7). Instead, we show that for a particular choice of (α,γ,δ), the remaining parameters (β,ρ) can be chosen such that the SDP is feasible, where the matrix in (7b) is rank one. We have performed extensive numerical optimizations of the SDP, suggesting that the optimal parameters do in fact have this structure. We now state our main design result, which describes the convergence rate of the SVL algorithm. We prove the result in Appendix C.

Theorem 12 (SVL): Consider applying Algorithm 1 to the distributed optimization problem (1), and suppose Assumptions 1 and 2 hold with $0 < m < L$ and $0 \le \sigma < 1$. Define $\eta := 1 + \rho - \kappa(1-\rho)$ and choose the parameters

$$\alpha = \frac{1-\rho}{m}, \qquad \gamma = 1+\beta, \qquad \delta = 1, \tag{11}$$

where β and $\rho \in \left[\frac{L-m}{L+m}, 1\right)$ satisfy the constraints

$$\big(2\beta - (1-\rho)(\kappa+1)\big)(\beta - 1 + \rho^2) < 0, \tag{12a}$$

$$\rho^2 \left(\frac{\beta - 1 + \rho^2}{\beta - 1 + \rho}\right)\left(\frac{2 - \eta - 2\beta}{2\rho^2\beta - (1-\rho^2)\eta}\right) \times \left(\frac{(2\rho^2+\eta)\beta - (1-\rho^2)\eta}{(1+\rho)(\eta - 2\eta\rho + 2\rho^2) - (2\rho^2+\eta)\beta}\right) = \sigma^2. \tag{12b}$$

Then there exists a constant $c > 0$ independent of i and k such that for all agents $i \in \{1,\dots,n\}$ and all iterations $k \ge 0$, $\|y_i^k - y_{\text{opt}}\| \le c\,\rho^k$ where $y_{\text{opt}} \in \mathbb{R}^d$ is the optimizer of (1).

Theorem 12 provides conditions on parameters $(\alpha,\beta,\gamma,\delta)$ of Algorithm 1 such that the algorithm converges with rate no slower than ρ. The theorem, however, does not address the problem of optimizing the convergence rate since β and ρ must only be chosen to satisfy the constraints (12). This is because the optimal parameters do not admit a closed-form solution for the convergence rate ρ as a function of the spectral gap σ and function parameters m and L. However, we now provide a systematic method for computing the optimal parameters.

The parameters must satisfy (12b), but this equation does not have a closed-form solution for ρ. Instead, we consider fixing the rate ρ and maximizing the corresponding spectral gap. We can then choose β to maximize σ^2 in (12b). Setting the derivative equal to zero, we find that the value of β which maximizes σ^2 for a fixed convergence rate ρ satisfies

$$\frac{\mathrm{d}\sigma^2}{\mathrm{d}\beta} = 0 \quad \Longrightarrow \quad \big(\beta\big(1-\kappa+2\rho(1+\rho)\big) - \eta(1-\rho^2)\big) \times \big(s_0 + s_1\beta + s_2\beta^2 + s_3\beta^3\big) = 0,$$

where the coefficients s_i are given by

$$s_0 := \eta\left(1-\rho^2\right)^2\big(\eta - (3-\eta)\eta\rho + 2(1-\eta)\rho^2 + 2\rho^3\big),$$

$$s_1 := -\big(1-\rho^2\big)\Big(\eta^3\rho + 4\rho^5 - 2\eta\rho^2(2\rho^2+\rho-3) + \eta^2\left(4\rho^3 - 4\rho^2 - 6\rho + 3\right)\Big),$$

$$s_2 := 3\eta(1-\rho)^2(1+\rho)(2\rho^2+\eta),$$

$$s_3 := (2\rho^2+\eta)(2\rho^3-\eta).$$

Solving the first factor for β, we find that it does not satisfy the inequality (12a) and is therefore not a valid solution. The optimal β must then make the second factor zero. Therefore, we can do a bisection search over ρ, where at each iteration of the bisection search we solve the cubic equation

$$s_0 + s_1\beta + s_2\beta^2 + s_3\beta^3 = 0 \tag{13}$$

to find the unique real solution β that satisfies (12a). Substituting this value of β into (12b) we can solve for σ. Denote the solution by $\hat{\sigma}$. If $\hat{\sigma} < \sigma$, we increase ρ; otherwise, we decrease ρ. We then repeat until $\hat{\sigma}$ is sufficiently close to σ. Refer to Algorithm 2 for a summary of this procedure

for finding the parameters β and ρ that optimize the worst-case convergence rate. We define SVL to be Algorithm 1 with parameters chosen using Algorithm 2.

Algorithm 2 (computing the SVL parameters)

Initialization: Let $0 < m < L$, $0 \leq \sigma < 1$, and $\varepsilon > 0$. Define $\kappa := L/m$. Set $\rho_1 = 0$ and $\rho_2 = 1$.
while $\rho_2 - \rho_1 > \varepsilon$ **do**
  $\rho = (\rho_1 + \rho_2)/2$
  Let β be the unique real solution to (13) that satisfies (12a).
  Using this value of β, let $\hat{\sigma}$ denote the solution to (12b).
  if $\hat{\sigma} < \sigma$ **then**
    $\rho_1 = \rho$
  else
    $\rho_2 = \rho$
  end if
end while
return ρ, β

Fig. 1 displays the worst-case convergence rate ρ as a function of the spectral gap σ and the centralized gradient rate $\frac{\kappa-1}{\kappa+1}$ for SVL. One of the remarkable aspects of the SVL algorithm is that it actually achieves the same worst-case convergence rate as *centralized* gradient descent if the spectral gap is sufficiently small. In this case, there is sufficient mixing among the agents so that the convergence rate is limited by the difficulty of the optimization problem and not the problem of having agents agree on the solution (i.e., consensus). This corresponds to the horizontal lines for small values of σ in the top panel of Fig. 1. Viewed another way, the convergence rate is limited by the difficulty of the optimization problem when the problem is ill-conditioned (i.e., κ is large), which corresponds to the curves approaching the straight line at $\rho = \frac{\kappa-1}{\kappa+1}$ in the bottom panel of Fig. 1.

Remark 13 (Optimality): We conjecture that the SVL parameters $(\alpha, \beta, \gamma, \delta)$ produce the fastest worst-case convergence rate over all algorithms in the form of Algorithm 1 that is certifiable using Theorem 10. However, we make no formal claims of optimality of the SVL algorithm in this paper.

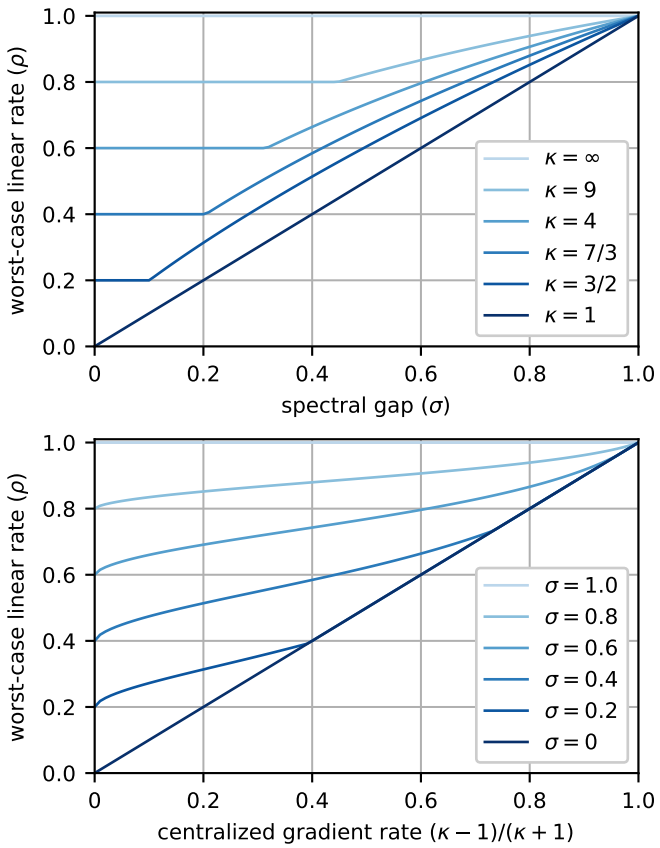


Fig. 1. Worst-case linear rate $\boldsymbol{\rho}$ of SVL in Theorem 12 as a function of $\boldsymbol{\kappa}$ and $\boldsymbol{\sigma}$. Top plot: as $\boldsymbol{\kappa} \to \mathbf{1}$ (quadratic objective), we obtain $\boldsymbol{\rho} = \boldsymbol{\sigma}$ (optimal linear consensus rate). Bottom plot: as $\boldsymbol{\sigma} \to \mathbf{0}$ (fully connected graph), we obtain $\boldsymbol{\rho} = \frac{\boldsymbol{\kappa}-\mathbf{1}}{\boldsymbol{\kappa}+\mathbf{1}}$ (optimal centralized gradient rate).

B. Interpretation of SVL as Inexact ADMM

To motivate the structure of SVL, we show how SVL can be interpreted as an inexact version of the alternating direction method of multipliers (ADMM). Using the formulation in [4, Section 7.1], the problem (1) can be solved using ADMM:

$$x_i^{k+1} = \arg\min_x f_i(x) + (x - y_i^k)^\mathsf{T} z_i^k + \tfrac{\beta}{2}\|x - y_i^k\|^2 \quad (14a)$$

$$y_i^{k+1} = \frac{1}{n}\sum_{j=1}^n x_j^{k+1} \quad (14b)$$

$$z_i^{k+1} = z_i^k + \beta\,(x_i^{k+1} - y_i^{k+1}) \quad (14c)$$

where (x_i^k, y_i^k, z_i^k) are the variables associated with agent i at time k, and β is the ADMM parameter. To implement this algorithm, however, each agent must solve the local optimization problem (14a) *exactly* as well as compute the *exact* average (14b) at each iteration. Instead, we consider a variant where the computations and communications are *inexact*. Specifically, we replace the exact minimization (14a) with a single gradient step with initial condition y_i^k and stepsize $\alpha > 0$, and we replace the exact averaging step (14b) with a single gossip step using the Laplacian matrix $\mathcal{L}^k$. This gives the following inexact version of ADMM:

$$x_i^{k+1} = y_i^k - \alpha\left(\nabla f(y_i^k) + z_i^k\right)$$

$$y_i^{k+1} = x_i^{k+1} - \sum_{j=1}^n \mathcal{L}_{ij}^{k+1} x_j^{k+1}$$

$$z_i^{k+1} = z_i^k + \beta\,(x_i^{k+1} - y_i^{k+1})$$

Defining the state $w_i^k := -\frac{\alpha}{\beta} z_i^{k-1}$, this algorithm is equivalent to Algorithm 1 with $\gamma = 1+\beta$ and $\delta = 1$. In other words, SVL corresponds to an inexact version of ADMM, where α is the stepsize of the gradient step and β is the ADMM parameter. See [5], [24] for other distributed ADMM variants.

C. Special Cases

We now show how the SVL algorithm reduces to well-known consensus and optimization algorithms in special cases.

a) $n = 1$: With only one agent, the distributed optimization problem (1) is equivalent to centralized optimization. In

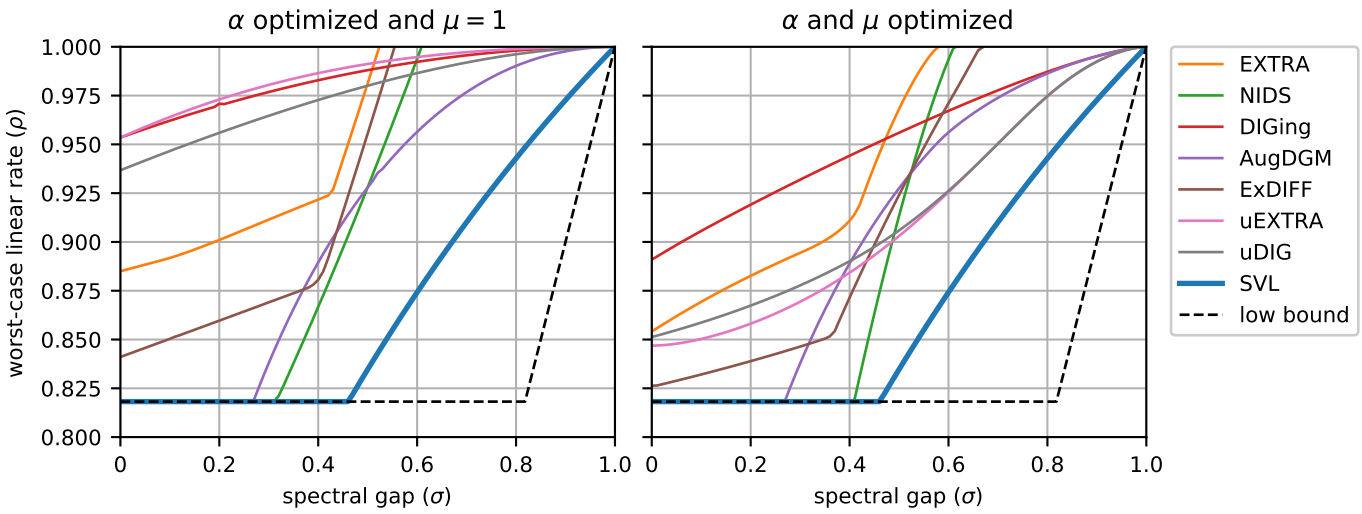


Fig. 2. Comparison of upper bounds for linear convergence rate ρ (smaller is better) as a function of graph connectedness σ, derived from Theorem 10 using $\kappa = 10$. (Left) stepsize α is optimized for each algorithm. (Right) both stepsize α and overelaxation parameter μ are optimized for each algorithm. The SVL algorithm (derived in Section IV) outperforms all the tested methods. SVL has no tunable parameters so it is the same in both scenarios. The lower bound (see Section II-E) corresponds to $\rho \geq \frac{\kappa-1}{\kappa+1} \approx 0.818$ (optimal centralized gradient rate) and $\rho \geq \sigma$ (optimal average consensus rate).

this case, the Laplacian matrix is simply the scalar $\mathcal{L}^k = 0$, so $v_1^k = 0$ for all $k \geq 0$. Algorithm 1 then simplifies to

$$x_1^{k+1} = x_1^k - \alpha \nabla f(x_1^k), \qquad x_1^0 \text{ arbitrary},$$

which is ordinary gradient descent with stepsize α. The fastest possible gradient rate of $\rho = \frac{\kappa-1}{\kappa+1}$ is achieved when $\alpha = \frac{2}{L+m}$.

b) $\kappa = 1$: When the condition ratio is unity (i.e., $m = L$), the distributed optimization problem (1) is equivalent to average consensus. In this case, the parameters of SVL are simply $\alpha = \frac{1}{L}$, $\beta = 1$, $\gamma = 2$, and $\delta = 1$. Also, the objective functions are quadratic, so we may assume without loss of generality that they have the form $f_i^k(x) = \frac{L}{2}\|x - r_i^k\|^2$, where $r_i^k \in \mathbb{R}^d$ is a parameter on agent $i \in \{1, \dots, n\}$ at iteration k. The SVL algorithm then simplifies to

$$x_i^{k+1} = x_i^k - \sum_{j=1}^{n} \mathcal{L}_{ij}^k x_j^k + \left(r_i^k - r_i^{k-1}\right), \qquad x_i^0 = r_i^0,$$

which is a dynamic average consensus algorithm since the reference signals are continually injected into the dynamics [10]. When the objective functions are constant, the r_i terms cancel from the iterations and only affect the initial conditions. This case is referred to as *static* average consensus [27], and the worst-case rate of convergence is $\rho = \sigma$ [29].

V. Numerical Results

In this section, we compare the worst-case performance of SVL with that of other first-order distributed algorithms.

A. Algorithm Comparison (Upper Bounds)

Theorem 10 provides an upper bound on the worst-case convergence rate. We used this result to compare all algorithms in Table I, including SVL. The results are shown in Fig. 2. For each algorithm, we used a bisection search to find the smallest rate ρ that yielded a feasible solution to the SDP (7). We implemented the SDP in Julia [3] with the JuMP [6] modeling package and the Mosek interior point solver [1]. In an outer loop, we performed a parameter search for each algorithm to find the step size α and overrelaxation parameter μ that yielded the smallest possible ρ. Specifically, we used Brent's method and the Nelder–Mead method, respectively, as implemented in the Optim package [14] as σ ranged from 0 to 1.

As shown in Fig. 2, optimizing over μ further improves worst-case performance. Our proposed SVL algorithm outperforms all methods we tested. Also shown in Fig. 2 is the lower bound described in Section II-E, namely $\rho \geq \max\{\frac{\kappa-1}{\kappa+1}, \sigma\}$, which holds for any distributed algorithm.

B. Approximate Worst-Case Examples (Lower Bounds)

In an effort to show that the upper bounds for each algorithm in Fig 2 were likely tight, we searched for signals $\{x^k, u^k, v^k, y^k, z^k\}$ that satisfied (3) for some choice of f_i and $\mathcal{L}^k$ satsifying Assumptions 1 and 2, respectively.

We first solved a relaxed version of the problem, where we replaced Assumptions 1 and 2 by the weaker conditions (17) and (18), respectively. We used the following greedy heuristic. For a given algorithm and rate ρ, we solved (7) to obtain (P, Q, R). At each time step k, we then maximized the Lyapunov increment $V^{k+1} - \rho^2 V^k$, where V^k is defined in (9). We solved the following optimization problem for $k \geq 0$.

$$\begin{aligned} \underset{u_i^k, v_i^k \in \mathbb{R}^d}{\text{maximize}} \quad & V^{k+1} - \rho^2 V^k \\ \text{such that} \quad & \text{(3a), (3c), and (18) hold,} \\ & \text{(17) holds for } i = 1, \dots, n, \\ & 1^{\mathsf{T}} v^k = 0. \end{aligned} \tag{15}$$

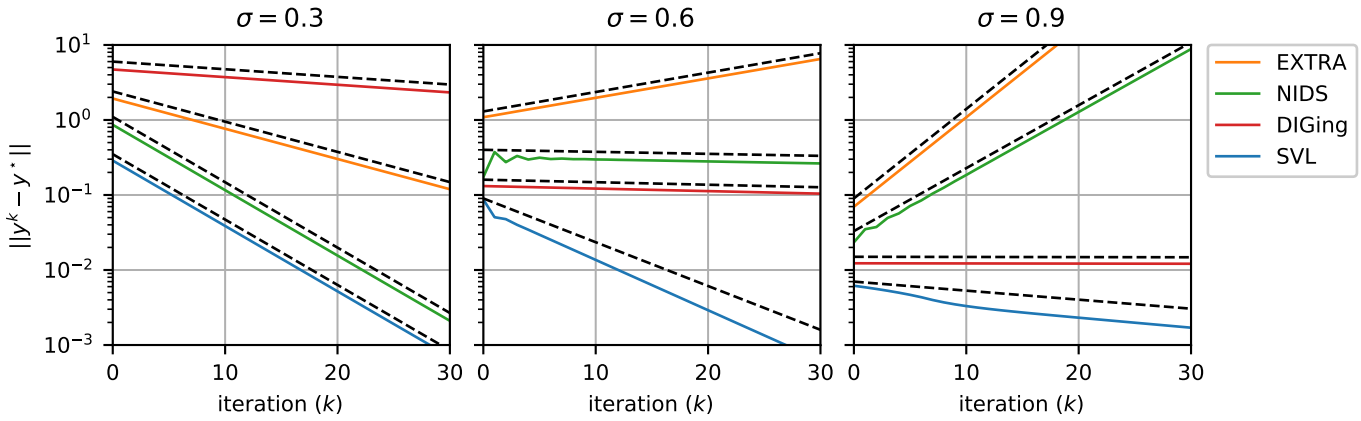


Fig. 3. Approximate worst-case trajectories for EXTRA, NIDS, DIGing, and SVL. Trajectories were found by solving the relaxed problem (15). We used $\boldsymbol{\alpha}$ optimized as in Fig. 2 and the default $\boldsymbol{\mu} = \mathbf{1}$. Simulations were performed for $\boldsymbol{\kappa} = \mathbf{10}$, $\boldsymbol{\sigma} \in \{\mathbf{0.3}, \mathbf{0.6}, \mathbf{0.9}\}$, and $\boldsymbol{n} = \boldsymbol{d} = \mathbf{2}$. Dashed lines indicate corresponding upper bounds obtained from Theorem 10 and shown in Fig. 2. All traces were vertically translated to improve clarity.

For $k = 0$, we also included x^0 as an optimization variable and the normalization $V^0 = 1$. For $k \geq 1$, we solved (15) using the x^k found at the previous iteration and warm-starting u^k, v^k. We used the Ipopt [28] local solver with default settings since (15) is a nonconvex quadratically constrained quadratic program. Note that we must choose parameters n and d.

Our relaxed heuristic using $n = d = 2$ was successful in constructing trajectories that matched the worst-case bounds from (7). To illustrate, we simulated EXTRA, NIDS, DIGing, and SVL with $\kappa = 10$ and a few values of σ in Fig. 3. For each trajectory, we plotted $\|y^k - y^\star\|$ together with the corresponding upper bound ρ found from Theorem 10. We obtained similar results for the other algorithms from Table I.

Since we used the relaxation (18) to construct z^k and v^k, there is no guarantee that there will exist a *linear* Laplacian $\mathcal{L}^k$ such that $v^k = \mathcal{L}^k z^k$. However, finding whether such an $\mathcal{L}^k$ exists amounts to solving a convex optimization problem:

$$\begin{array}{ll} \underset{\mathcal{L}^k \in \mathbb{R}^{n\times n}}{\text{minimize}} & \|I - \Pi - \mathcal{L}^k\| \\ \text{such that} & (\mathcal{L}^k \otimes I) z^k = v^k, \\ & \mathcal{L}^k 1 = 0, \quad 1^\mathsf{T} \mathcal{L}^k = 0. \end{array} \tag{16}$$

If (16) is feasible and its optimal value is less than or equal to σ, then the associated $\mathcal{L}^k$ is a valid Laplacian matrix at timestep k. While there is no guarantee that (16) will even be feasible, we reasoned that since there are n^2 variables and $2n + ndc$ linear constraints, where d and c are the number of rows of C_y and C_z, respectively, we could increase our chances of finding feasible $\mathcal{L}^k$ with n large and d and c small.

In Figure 4, we show a successful construction for the NIDS algorithm, which has $c = 1$. We solved (15) with $n = 15$ and $d = 1$, and solved (16) at each timestep. An optimal cost for (16) of σ was always achieved.

This result indicates that the upper bound for NIDS in Fig. 2 is likely tight, and that NIDS is not robustly stable in the time-varying setting. In other words, the *network-independent* rate bound enjoyed by NIDS in the constant-graph setting [13, Thm. 2] does not carry over to the time-varying setting.

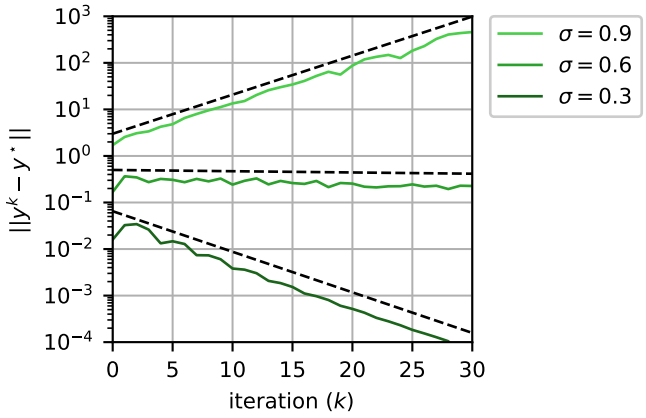


Fig. 4. Worst-case trajectories for NIDS found by solving (15) and successfully solving (16) to construct a sequence of Laplacians $\{\mathcal{L}^k\}$. Simulations were performed using optimized $\boldsymbol{\alpha}$, $\boldsymbol{\mu} = \mathbf{1}$, $\boldsymbol{\kappa} = \mathbf{10}$, $\boldsymbol{n} = \mathbf{15}$, and $\boldsymbol{d} = \mathbf{1}$ for $\boldsymbol{\sigma} \in \{\mathbf{0.3}, \mathbf{0.6}, \mathbf{0.9}\}$. Trajectories were plotted with their accompanying rate bounds (dashed lines) from Theorem 10 and translated to improve clarity.

Remark 14: There may be other approaches to finding a worst-case $\mathcal{L}^k$ that perform better. For example, one might try alternating convex optimizations or including $\mathcal{L}^k$ directly as an optimization variable in a nonlinear program.

VI. Conclusion

We presented a universal analysis framework for a broad class of first-order distributed optimization algorithms over time-varying graphs. The framework provides worst-case certificates of linear convergence via semidefinite programming, and we show empirically that our rate bounds are likely tight. Optimizing the SDP from our analysis framework, we designed a novel distributed algorithm, SVL, which outperforms all known algorithms in this time-varying setting.

REFERENCES

[1] APS Mosek. The MOSEK optimization software, 2010. Online at http://www.mosek.com.

[2] J. A. Bazerque and G. B. Giannakis. Distributed spectrum sensing for cognitive radio networks by exploiting sparsity. *IEEE Transactions on Signal Processing*, 58(3):1847–1862, 2009.

[3] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, volume 3. Foundations and Trends in Machine Learning, 2010.

[5] T. Chang, M. Hong, and X. Wang. Multi-agent distributed optimization via inexact consensus admm. *IEEE Transactions on Signal Processing*, 63(2):482–497, 2015.

[6] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[7] P. A. Forero, A. Cano, and G. B. Giannakis. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11:1663–1707, 2010.

[8] D. Jakovetić. A unification and generalization of exact distributed first-order methods. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):31–46, 2018.

[9] B. Johansson. *On distributed optimization in networked systems*. PhD thesis, KTH, 2008.

[10] S. S. Kia, B. Van Scoy, J. Cortés, R. A. Freeman, K. M. Lynch, and S. Martínez. Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. *IEEE Control Systems Magazine*, 39(3):40–72, 2019.

[11] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[12] L. Lessard and P. Seiler. Direct synthesis of iterative algorithms with bounds on achievable worst-case convergence rate. *American Control Conference*, 2020.

[13] Z. Li, W. Shi, and M. Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019.

[14] P. K. Mogensen and A. N. Riseth. Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24), 2018.

[15] A. Nedić, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[16] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004.

[17] J. Ortega and W. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic press, 1970.

[18] G. Qu and N. Li. Accelerated distributed Nesterov gradient descent for smooth and strongly convex functions. In *Allerton Conference on Communication, Control, and Computing*, pages 209–216, 2016.

[19] G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 2017.

[20] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 20–27. ACM, 2004.

[21] S. S. Ram, V. V. Veeravalli, and A. Nedić. Distributed non-autonomous power control through distributed convex optimization. In *IEEE INFOCOM*, pages 3001–3005, 2009.

[22] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3027–3036, 2017.

[23] W. Shi, Q. Ling, G. Wu, and W. Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[24] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

[25] A. Sundararajan, B. Hu, and L. Lessard. Robust convergence analysis of distributed optimization algorithms. In *Allerton Conference on Communication, Control, and Computing*, pages 1206–1212, 2017.

[26] A. Sundararajan, B. Van Scoy, and L. Lessard. A canonical form for first-order distributed optimization algorithms. In *American Control Conference*, pages 4075–4080, 2019.

[27] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, Massachusetts Institute of Technology, 1984.

[28] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[29] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[30] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing*, 67(1):33–46, 2007.

[31] R. Xin, D. Jakovetić, and U. A. Khan. Distributed nesterov gradient methods over arbitrary graphs. *IEEE Signal Processing Letters*, 2019.

[32] R. Xin and U. A. Khan. A linear algorithm for optimization over directed graphs with geometric convergence. *IEEE Control Systems Letters*, 2(3):315–320, 2018.

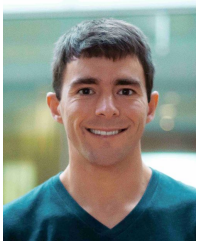
[33] R. Xin and U. A. Khan. Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking. *IEEE Transactions on Automatic Control*, 2019.

[34] J. Xu, S. Zhu, Y. C. Soh, and L. Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *IEEE Conference on Decision and Control*, pages 2055–2060, 2015.

[35] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed. Exact diffusion for distributed optimization and learning—Part I: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3):708–723, 2018.

[36] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed. Exact diffusion for distributed optimization and learning—Part II: Convergence analysis. *IEEE Transactions on Signal Processing*, 67(3):724–739, 2018.

Akhil Sundararajan received the B.S. in electrical engineering and computer sciences and materials science and engineering from the University of California, Berkeley, and the M.S. in electrical and computer engineering from the University of Wisconsin–Madison. He is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of Wisconsin–Madison.



Bryan Van Scoy (M'13) is a postdoc at the Wisconsin Institute for Discovery at the University of Wisconsin–Madison. He received the Ph.D. degree in electrical engineering and computer science from Northwestern University in 2017, and B.S. and M.S. degrees in applied mathematics along with the B.S.E. in electrical engineering from the University of Akron in 2012. His research interests include multi-agent systems, convex optimization, and robust control.

Laurent Lessard (M'09) was born in Toronto, Canada. He received the BASc in engineering science from the University of Toronto and the M.S. and Ph.D. degrees in aeronautics and astronautics from Stanford University.

He is a Charles Ringrose Assistant Professor of Electrical and Computer Engineering at the University of Wisconsin–Madison and a faculty member of the Wisconsin Institute for Discovery. Previously, he was a postdoc in the Berkeley Center for Control and Identification at the University of California, Berkeley, and a LCCC postdoc in the Department of Automatic Control at Lund University.

Dr. Lessard received the NSF CAREER Award in 2018 and the O. Hugo Schuck Best Paper Award in 2013.

APPENDIX

A. Proof of Proposition 6

Suppose (6) holds, and denote the optimizer of (1) by y_{opt}. Then there exist vectors p and q such that

$$\begin{cases} 0 = (A-I)\,p \\ y_{\text{opt}} = C_y p \\ 0 = F_x p \end{cases} \quad \text{and} \quad \begin{cases} B_u = (A-I)\,q \\ D_{yu} = C_y q \\ D_{zu} = C_z q. \end{cases}$$

For all $i \in \{1,\dots,n\}$, use these vectors to define the points

$$\begin{aligned} x_i^\star &= p - q\,\nabla f_i(y_{\text{opt}}), & y_i^\star &= y_{\text{opt}}, & z_i^\star &= C_z p, \\ u_i^\star &= \nabla f_i(y_{\text{opt}}), & v_i^\star &= 0. \end{aligned}$$

This is a fixed point of algorithm (3), and the fixed point satisfies the conditions in (5) since y_{opt} is the optimizer of (1).

Now suppose $(x^\star, y^\star, z^\star, u^\star, v^\star)$ is a fixed point of (3) satisfying (5). Let $p = (1/n)\sum_{i=1}^n x_i^\star$. Since $1^\mathsf{T} u^\star = 0$, $v^\star = 0$, and $1^\mathsf{T} y^\star$ is unconstrained, we have from (3a) and (3c) that $p \neq 0$ is in the set (6a). Now let r be any nonzero vector such that $r^\mathsf{T} 1 = 0$. Then from (3a), we have that

$$0 = \begin{bmatrix} A-I \\ C_y \\ C_z \end{bmatrix} (r^\mathsf{T} x^\star) + \begin{bmatrix} B_u \\ D_{yu} \\ D_{zu} \end{bmatrix} (r^\mathsf{T} u^\star).$$

Since this must hold for arbitrary $r^\mathsf{T} u^\star$, this implies (6b). ∎

B. Proof of Theorem 10

Assumptions 1 and 2 lead to quadratic inequalities that will be useful in proving our main result. These are stated in the following propositions.

Proposition 15: Suppose Assumption 1 holds for the local objective functions f_i. Let (y_i^k, u_i^k) satisfy (3b), and let $(y_i^\star, u_i^\star)$ be a fixed point that satisfies (5). Then

$$\begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^\mathsf{T} (M_0 \otimes I) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} \geq 0. \tag{17}$$

Proof: Using the definition of M_0, the quadratic form is

$$\begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^\mathsf{T} (M_0 \otimes I) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} = -2\sum_{i=1}^n (\tilde{u}_i^k - m\tilde{y}_i^k)^\mathsf{T} (\tilde{u}_i^k - L\tilde{y}_i^k).$$

Since the fixed point satisfies (5), Assumption 1 implies that this is nonnegative with $y_{\text{opt}} = y_1^\star = \dots = y_n^\star$. ∎

Proposition 16: Suppose Assumption 2 holds for the graph $\mathcal{G}^k$ at each iteration. Let (z_i^k, v_i^k) satisfy (3b), and let $(z_i^\star, v_i^\star)$ be a fixed point that satisfies (5). Then for all $R \succeq 0$,

$$\begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix}^\mathsf{T} \big(M_1 \otimes (I-\Pi) \otimes R\big) \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix} \geq 0. \tag{18}$$

Proof: From the definition of the matrix norm and Assumption 2, we have that

$$\begin{aligned} \sigma \geq \left\| I - \Pi - \mathcal{L}^k \right\| &= \left\| (I - \Pi - \mathcal{L}^k)(I-\Pi) \right\| \\ &= \max_{y \in \mathbb{R}^n, y \neq 0} \frac{\left\| (I - \Pi - \mathcal{L}^k)(I-\Pi) y \right\|}{\|y\|}. \end{aligned}$$

Without loss of generality, $y = \Pi\eta + (I-\Pi)\,\phi$, where η and ϕ are arbitrary. By orthogonality, $\|y\|^2 = \|\Pi\eta\|^2 + \|(I-\Pi)\,\phi\|^2$. Substituting the decomposition of y into the above inequality,

$$\begin{aligned} \sigma &\geq \max_{\phi,\eta \in \mathbb{R}^n, y \neq 0} \frac{\left\| (I - \Pi - \mathcal{L}^k)(I-\Pi)\,\phi \right\|}{\sqrt{\|\Pi\eta\|^2 + \|(I-\Pi)\,\phi\|^2}} \\ &= \max_{\phi \in \mathbb{R}^n, y \neq 0} \frac{\left\| (I - \Pi - \mathcal{L}^k)(I-\Pi)\,\phi \right\|}{\|(I-\Pi)\,\phi\|} \\ &= \max_{\phi \in \mathbb{R}^n, y \neq 0} \frac{\left\| (I-\Pi)(\phi - \mathcal{L}^k \phi) \right\|}{\|(I-\Pi)\,\phi\|}, \end{aligned}$$

where the last two steps follow because the maximum is attained with $\eta = 0$, and $\mathcal{L}^k \Pi = \Pi \mathcal{L}^k = \mathbf{0}$. Squaring both sides and rewriting as a quadratic form yields

$$\begin{bmatrix} \phi \\ \mathcal{L}^k \phi \end{bmatrix}^\mathsf{T} \big(M_1 \otimes (I-\Pi)\big) \begin{bmatrix} \phi \\ \mathcal{L}^k \phi \end{bmatrix} \geq 0 \tag{19}$$

for all $\phi \in \mathbb{R}^n$. Now let p denote the dimension of z_i^k. Then since $R \succeq 0$, it has the decomposition

$$R = \sum_{\ell=1}^p \mu_\ell\, w_\ell w_\ell^\mathsf{T},$$

where $w_\ell \in \mathbb{R}^p$ and $\mu_\ell \geq 0$. Then using that $\tilde{v}^k = (\mathcal{L}^k \otimes I_p)\,\tilde{z}^k$, the quadratic form is

$$\begin{aligned} &\begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix}^\mathsf{T} \big(M_1 \otimes (I-\Pi) \otimes R\big) \begin{bmatrix} \tilde{z}^k \\ \tilde{v}^k \end{bmatrix} \\ &\quad = \sum_\ell \mu_\ell \begin{bmatrix} \star \end{bmatrix}^\mathsf{T} \big(M_1 \otimes (I-\Pi)\big) \begin{bmatrix} (I \otimes w_\ell^\mathsf{T})\,\tilde{z}^k \\ (I \otimes w_\ell^\mathsf{T})\,\tilde{v}^k \end{bmatrix} \\ &\quad = \sum_\ell \mu_\ell \begin{bmatrix} \star \end{bmatrix}^\mathsf{T} \big(M_1 \otimes (I-\Pi)\big) \begin{bmatrix} (I \otimes w_\ell^\mathsf{T})\,\tilde{z}^k \\ \mathcal{L}^k\,(I \otimes w_\ell^\mathsf{T})\,\tilde{z}^k \end{bmatrix}, \end{aligned}$$

which is nonnegative from (19) with $\phi \leftarrow (I \otimes w_\ell^\mathsf{T})\,\tilde{z}^k$. ∎

Let $(x^k, y^k, z^k, u^k, v^k)$ denote a trajectory of algorithm (3). Since the algorithm satisfies the fixed point conditions (6) (by assumption), we have from Proposition 6 that there exists a fixed point $(x^\star, y^\star, z^\star, u^\star, v^\star)$ satisfying (5). The global optimizer is unique from Assumption 1, so the fixed point conditions (5a) imply that $y_1^\star = \dots = y_n^\star = y_{\text{opt}}$ with y_{opt} the optimizer of (1).

Since the trajectory satisfies the invariant (3c) and the columns of Ψ form a basis for the nullspace of $\begin{bmatrix} F_x & F_u \end{bmatrix}$, there exists a vector $\tilde{s}^k$ such that

$$\Psi\,\tilde{s}^k = \frac{1}{\sqrt{n}} \sum_{i=1}^n \begin{bmatrix} \tilde{x}_i^k \\ \tilde{u}_i^k \end{bmatrix}.$$

Multiplying the matrix in (7a) on the right and left by $\tilde{s}^k$ and its transpose, respectively, we obtain the consensus inequality

$$\begin{aligned} &(\tilde{x}^{k+1})^\mathsf{T} (\Pi \otimes P)\,\tilde{x}^{k+1} - \rho^2\,(\tilde{x}^k)^\mathsf{T} (\Pi \otimes P)\,\tilde{x}^k \\ &\qquad + \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix}^\mathsf{T} (M_0 \otimes \Pi) \begin{bmatrix} \tilde{y}^k \\ \tilde{u}^k \end{bmatrix} \leq 0. \end{aligned} \tag{20a}$$

Now choose the vectors $w_2, \dots, w_n \in \mathbb{R}^n$ such that the matrix $\begin{bmatrix} 1_n/\sqrt{n} & w_2 & \dots & w_n \end{bmatrix}$ is orthonormal. Then we can

multiply the matrix in (7b) on the right and left by the weighted sum

$$\sum_{i=1}^{n}(w_\ell)_i \begin{bmatrix}\tilde{x}_i^k\\ \tilde{u}_i^k\\ \tilde{v}_i^k\end{bmatrix}$$

and its transpose, respectively, and sum over $\ell \in \{2,\dots,n\}$ to obtain the disagreement inequality

$$\begin{aligned}(\tilde{x}^{k+1})^{\mathsf{T}}\big((I-\Pi)\otimes Q\big)\tilde{x}^{k+1} - \rho^2(\tilde{x}^k)^{\mathsf{T}}\big((I-\Pi)\otimes Q\big)\tilde{x}^k \\ + \begin{bmatrix}\tilde{y}^k\\ \tilde{u}^k\end{bmatrix}^{\mathsf{T}}\big(M_0\otimes(I-\Pi)\big)\begin{bmatrix}\tilde{y}^k\\ \tilde{u}^k\end{bmatrix} \\ + \begin{bmatrix}\tilde{z}^k\\ \tilde{v}^k\end{bmatrix}^{\mathsf{T}}\big(M_1\otimes(I-\Pi)\otimes R\big)\begin{bmatrix}\tilde{z}^k\\ \tilde{v}^k\end{bmatrix} \le 0, \quad (20b)\end{aligned}$$

where we used that $\{w_i\}_{i=1}^n$ form an orthonormal basis for $\mathbb{R}^n$. Summing the inequalities in (20), we obtain

$$\begin{aligned}V^{k+1} - \rho^2 V^k + \begin{bmatrix}\tilde{y}^k\\ \tilde{u}^k\end{bmatrix}^{\mathsf{T}}(M_0\otimes I)\begin{bmatrix}\tilde{y}^k\\ \tilde{u}^k\end{bmatrix} \\ + \begin{bmatrix}\tilde{z}^k\\ \tilde{v}^k\end{bmatrix}^{\mathsf{T}}\big(M_1\otimes(I-\Pi)\otimes R\big)\begin{bmatrix}\tilde{z}^k\\ \tilde{v}^k\end{bmatrix} \le 0,\end{aligned}$$

where V^k is defined in (9). The quadratic forms in the last two terms are nonnegative from Propositions 15 and 16, which implies $V^{k+1} \le \rho^2 V^k$. We then apply this inequality iteratively to obtain $V^k \le \rho^{2k} V_0$ for all $k \ge 0$. Now define

$$T := \Pi\otimes P + (I-\Pi)\otimes Q,$$

and note that $T \succ 0$ since P and Q are positive definite. Then letting $\mathrm{cond}(T) = \lambda_{\max}(T)/\lambda_{\min}(T)$ denote the condition number of T, we have the bound

$$\|x_i^k - x_i^\star\|^2 \le \|x^k - x^\star\|^2 \le \mathrm{cond}(T)\, V^k \le \rho^{2k}\,\mathrm{cond}(T)\, V^0.$$

Therefore, the bound (8) holds with $c = \sqrt{\mathrm{cond}(T)\, V^0}$. ∎

C. Proof of Theorem 12

Substituting the template (10) into the LMI (7a) reduces to

$$P_{11}\begin{bmatrix}1-\rho^2 & -\alpha\\ -\alpha & \alpha^2\end{bmatrix} + M_0 \preceq 0,$$

which is satisfied with $\alpha = (1-\rho)/m$ and $P_{11} = \frac{m(L-m)}{\rho(1-\rho)}$. Note that this LMI is known to describe the convergence rate of centralized gradient descent; see [11, Section 4.4].

Now consider the potential solution to (7b) given by

$$Q = \frac{t_3}{\alpha^2\rho^2}\begin{bmatrix}1+\rho^2\frac{t_1}{t_4} & -1\\ -1 & 1\end{bmatrix} \quad \text{and} \quad R = \frac{t_5}{\alpha^2 t_2}, \quad \text{where}$$

$$\begin{aligned}t_1 &:= 2(1-\beta)-\eta, & t_2 &:= \beta - 1 + \rho^2,\\ t_3 &:= \beta(\eta+2\rho^2) - \eta(1-\rho^2), & t_4 &:= 2\beta\rho^2 - \eta(1-\rho^2),\\ t_5 &:= (1-\beta-\rho)\big(\beta(\eta+2\rho^2) - (1-\rho^2)(1-\kappa+2\kappa\rho)\big),\\ t_6 &:= \big(2-\alpha(L+m)\big)(1-\rho^2)^2 - \big(2(1-\rho^4) - \alpha(L+m)\big)\beta.\end{aligned}$$

Using these values along with the value for σ^2 in (12b), the matrix in (7b) is equal to the rank-one matrix $-\frac{1}{t_2 t_4} z z^{\mathsf{T}}$, where

$$z := \frac{1}{\alpha\rho}\begin{bmatrix}t_6\\ -t_2 t_3\\ \alpha\, t_2\,\big(2-\alpha(L+m)\big)\\ \beta\,\big(t_3 - \alpha\rho^2(L+m)\big)\end{bmatrix}.$$

In order for this to be a valid solution, we must have $t_3 > 0$ and $t_1/t_4 > 0$ (so that $Q \succ 0$), $t_5/t_2 \ge 0$ (so that $R \succeq 0$), and $t_2 t_4 > 0$ (so that (7b) holds). All of these inequalities hold if and only if (12a) holds. Therefore, the SDP has a rank-one solution using the parameters in (11) if β and ρ satisfy (12). The convergence bound then follows from Theorem 10 and Remark 11. ∎