Systematic Design of Decentralized Algorithms for Consensus Optimization

Shuo Han

***Abstract*—We propose a separation principle that enables a systematic way of designing decentralized algorithms used in consensus optimization. Specifically, we show that a decentralized optimization algorithm can be constructed by combining a non-decentralized base optimization algorithm and decentralized consensus tracking. The separation principle provides modularity in both the design and analysis of algorithms under an automated convergence analysis framework using integral quadratic constraints (IQCs). We show that consensus tracking can be incorporated into the IQC-based analysis. The workflow is illustrated through the design and analysis of a decentralized algorithm based on the alternating direction method of multipliers.**

***Index Terms*—Optimization algorithms, robust control**

I. Introduction

IN THIS paper, we study algorithms for solving the *consensus optimization problem*, which has the form

$$\min_{x_0 \in \mathbb{R}^d} \quad f_0(x_0) \coloneqq \frac{1}{n} \sum_{i=1}^{n} f_i(x_0).$$

We assume $f_i \colon \mathbb{R}^d \to \mathbb{R}$ is convex for $i = 1, 2, \ldots, n$ and the set of minimizers is nonempty. The name consensus optimization is due to the fact that the problem can be made equivalent to another optimization problem with a separable objective function $\sum_{i=1}^{n} f_i(x_i)$ by introducing local optimization variables $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ and a consensus constraint $x_1 = x_2 = \cdots = x_n$.

We are interested in algorithms that solve the consensus optimization problem in a *decentralized* manner. We shall make a distinction between distributed and decentralized algorithms, which are often used interchangeably in the literature; the former permits the presence of a *master node* that collects computational results from multiple worker nodes, whereas the latter does not require a master node. Most existing decentralized algorithms used in consensus optimization belong to one of the following two classes. The first one is based on the gradient descent method or its variants (e.g., Nesterov's method). This includes, among others, the distributed gradient descent method [10] (which is, in fact, decentralized despite its name), DIGing [9], [14], and EXTRA [15]. See also [13] for an algorithm based on Nesterov's method and [21] for handling directed communication graphs. The second one is based on operator splitting methods, of which the most widely used is the Douglas–Rachford method [12] or its application to the dual problem, the alternating direction method of multipliers (ADMM) [1]. Although the original ADMM algorithm, when directly applied to the consensus optimization problem, requires a master node and therefore is not decentralized, it has been shown that ADMM can be made decentralized by reformulating the consensus constraint [16], [20].

Despite a vast body of literature on decentralized optimization algorithms in recent years, there has been little work on systematic understanding and designing of decentralized algorithms. As a result, whenever the base optimization algorithm changes (e.g., from regular gradient descent to accelerated gradient descent) or the conditions on the communication graph changes (e.g., from undirected to directed), a convergence analysis of the new algorithm needs to be started almost from scratch. This paper seeks a framework that enables a systematic design of decentralized optimization algorithms in the hope of speeding up the development of new algorithms.

We believe that such a framework can be made possible through the automated convergence analysis of optimization algorithms proposed recently by Lessard et al. [7] Unlike traditional proof-based analysis that needs to be carried out manually, their automated convergence analysis uses computational tools to establish a numerical certificate of convergence for optimization algorithms. The key is to view an optimization algorithm as a feedback interconnection of a linear system and a nonlinear uncertain system characterized by *integral quadratic constraints* (IQCs) [8]. In this view, convergence of optimization algorithms can be established by certifying stability of the feedback interconnection.

Contribution: The main contribution of this paper is a *separation principle* for the design of decentralized algorithms used in consensus optimization. Specifically, one can start with a non-decentralized base optimization algorithm and replace the (static) averaging operation therein with decentralized average consensus tracking, which is designed independently from the base algorithm. While a similar idea was previously conjectured in [14, Rem. 1], we show that the separation between the base algorithm and consensus tracking brings modularity and a number of associated benefits. Such an approach not only enables a systematic way of designing decentralized optimization algorithms but is also amenable to automated convergence analysis based on IQCs. We believe that the result will help unify existing decentralized algorithms and eventually facilitate the development of new algorithms.

When applied to known settings (i.e., same base algorithm, same conditions on the communication graph), the result of this paper is not guaranteed to yield a better (e.g., with faster convergence, or more robust) decentralized algorithm than existing ones; our main focus is a more principled design procedure rather than optimality.

The author is with the Department of Electrical and Computer Engineering, University of Illinois, Chicago, IL 60607. hanshuo@uic.edu.

II. Main Results

A. Notation

Denote by $\mathbf{1}$ the column vector of all ones, $\|\cdot\|$ the ℓ_2-norm of a vector, I_n the $n \times n$ identity matrix (size omitted when clear from the context), $\otimes$ the Kronecker product, and $\sigma_{\max}(\cdot)$ the maximum singular value of a matrix. We define $J \coloneqq \mathbf{1}\mathbf{1}^T/n$ and $J_\perp \coloneqq I - J$. We use exclusively W to denote a symmetric irreducible doubly stochastic matrix (called *gossip matrix* in the setting of consensus): $W\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^T W = \mathbf{1}^T$. For a symmetric matrix P, we write $P \succeq 0$ if P is positive semidefinite. For a differentiable function f, we denote by ∇f the gradient of f. We often use $*$ to denote objects that can be inferred from symmetry.

In decentralized optimization, each node keeps its own local variables. We reserve the subscript for indexing the nodes and the superscript for indexing a given sequence. For example, $x_i \in \mathbb{R}^d$ represents a local variable that belongs to node i, whereas a sequence is denoted by $\{x^k\}_{k\geq 0} \coloneqq \{x^0, x^1, \dots\}$. We sometimes use a second superscript as in, e.g., $(x^{1,k}, x^{2,k})$ to index components in a tuple. For any convergent sequence $\{x^k\}$, we use $x^\star$ to denote its limit or, alternatively, steady-state value. We use the notation

$$x \coloneqq \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T \in \mathbb{R}^{n\times d}$$

to denote the matrix whose rows are formed by local variables $x_1, x_2, \dots, x_n \in \mathbb{R}^d$, and we define $\mathsf{ave}(x) \coloneqq \frac{1}{n}\sum_{i=1}^n x_i = \frac{1}{n} x^T \mathbf{1} \in \mathbb{R}^d$. Similarly, we use the notation

$$\nabla f(x) \coloneqq \begin{bmatrix} \nabla f_1(x_1) & \cdots & \nabla f_n(x_n) \end{bmatrix}^T \in \mathbb{R}^{n\times d}$$

to denote the list of local gradients.

B. Problem description

We investigate algorithms that solve the consensus optimization problem and can be expressed in one of the following two forms.

1) Centralized algorithms:

$$\xi_0^{k+1} = A_0 \xi_0^k + B_0\, \mathsf{ave}(u^k) \tag{1a}$$
$$v_0^k = C_0 \xi_0^k + D_0\, \mathsf{ave}(u^k) \tag{1b}$$
$$u_i^k = \phi_i(v_0^k), \quad i = 1, 2, \dots, n. \tag{1c}$$

2) Distributed algorithms: for $i = 1, 2, \dots, n$,

$$\xi_i^{k+1} = A_i \xi_i^k + B_{\mathrm{loc},i} u_i^k + B_i\, \mathsf{ave}(u^k) \tag{2a}$$
$$v_i^k = C_i \xi_i^k + D_{\mathrm{loc},i} u_i^k + D_i\, \mathsf{ave}(u^k) \tag{2b}$$
$$u_i^k = \phi_i(v_i^k). \tag{2c}$$

In both cases, ϕ_i is a continuous but possibly nonlinear function; A_i, B_i, C_i, D_i, $B_{\mathrm{loc},i}$, and $D_{\mathrm{loc},i}$ are all constant matrices of appropriate dimensions. (The subscript "loc" stands for *local*.) We give for each case one example algorithm that solves the consensus optimization problem. To simplify notation, we assume in the remaining part of this section that each f_i is smooth, but we expect the result to generalize to nonsmooth objective functions via the use of subdifferentials.

Example 1 (Gradient descent)**.** When applied to consensus optimization, the gradient descent algorithm becomes

$$x_0^{k+1} = x_0^k - \eta \nabla f_0(x_0^k) = x_0^k - \frac{\eta}{n}\sum_{i=1}^n \nabla f_i(x_0^k),$$

where $\eta > 0$ is a constant. Define $\xi_0^k \coloneqq x_0^k$, $v_0^k \coloneqq \xi_0^k$, and $u_i^k \coloneqq \nabla f_i(v_0^k)$. The gradient descent algorithm can be written as

$$\xi_0^{k+1} = \xi_0^k - \eta\, \mathsf{ave}(u^k), \qquad v_0^k = \xi_0^k.$$

Example 2 (ADMM)**.** When applied to consensus optimization, the ADMM algorithm becomes [1, p. 50]

$$x_i^{k+1} = \mathsf{ave}(x^k) - (y_i^k + w_i^k)/\rho$$
$$y_i^{k+1} = \mathsf{ave}(w^k) - w_i^k$$
$$w_i^k = \nabla f_i(x_i^{k+1}),$$

where $\rho > 0$ is a constant. Define $\xi_i^k \coloneqq (x_i^k, y_i^k)$, $v_i^k = (v_i^{1,k}, v_i^{2,k}) \coloneqq (x_i^k, x_i^{k+1})$, and $u_i^k = (u_i^{1,k}, u_i^{2,k}) \coloneqq (v_i^{1,k}, \nabla f_i(v_i^{2,k}))$. The ADMM algorithm can be written as

$$\xi_i^{k+1} = \begin{bmatrix} 0 & -\frac{I}{\rho} \\ 0 & 0 \end{bmatrix} \xi_i^k + \begin{bmatrix} -\frac{I}{\rho} \\ -I \end{bmatrix} u_i^{2,k} + \begin{bmatrix} \mathsf{ave}(u^{1,k}) \\ \mathsf{ave}(u^{2,k}) \end{bmatrix}$$
$$v_i^k = \begin{bmatrix} I & 0 \\ 0 & -\frac{I}{\rho} \end{bmatrix} \xi_i^k + \begin{bmatrix} 0 \\ -\frac{I}{\rho} \end{bmatrix} u_i^{2,k} + \begin{bmatrix} 0 \\ \mathsf{ave}(u^{1,k}) \end{bmatrix}.$$

The algorithms given in (1) and (2) are not fully decentralized because evaluating ave requires a master node to collect information from all the nodes. Moreover, in the first case of centralized algorithms, the computation in (1a) and (1b) needs to be completed by the master node as well and therefore is not decentralized either. Our goal in this paper is to develop a systematic procedure for converting an existing algorithm of the form (1) or (2) into a decentralized algorithm.

C. Main results

The key component in our procedure of decentralization is consensus tracking. We say that a dynamical system G_{con} achieves *average consensus tracking* (or simply *consensus tracking*) if for any sequence $s = \{s^k \in \mathbb{R}^{n\times d}\}$ converging to $s^\star$, the output $\hat{s} = G_{\mathrm{con}} s$ converges to $Js^\star$. An example of a system that achieves consensus tracking (used in the DIGing algorithm [9], [14]) is given by

$$\hat{s}^{k+1} = W\hat{s}^k + (s^{k+1} - s^k), \qquad \hat{s}^0 = s^0. \tag{3}$$

Systems that achieve consensus tracking are not unique. For example, the system in (3) can be modified slightly as

$$\hat{s}^{k+1} = W(\hat{s}^k + s^{k+1} - s^k), \qquad \hat{s}^0 = Ws^0, \tag{4}$$

which can be shown to also achieve consensus tracking.

The main idea behind converting an algorithm of the form (1) or (2) into a decentralized one is to replace the ave operator with a system G_{con} that achieves consensus tracking. In addition, the computation in (1a) and (1b) also needs to be decentralized, which can be handled by consensus tracking as well. For centralized algorithms of the form (1), the resulting decentralized algorithm after conversion is described in the theorem below and also illustrated in Fig. 1.

Theorem 3. *Suppose G_{con} is a system that achieves consensus tracking, and $\xi_0^\star$ is a possible steady-state value of ξ_0 in (1). Then, $\xi_0^\star$ is also a steady-state value of ξ_i in*

$$\xi_i^{k+1} = A_0\xi_i^k + B_0\hat{u}_i^k \tag{5a}$$
$$v_i^k = C_0\xi_i^k + D_0\hat{u}_i^k \tag{5b}$$
$$u_i^k = \phi_i(\hat{v}_i^k) \tag{5c}$$

for $i = 1, 2, \dots, n$, where $\hat{u} = G_{\text{con}}u$ and $\hat{v} = G_{\text{con}}v$.

Proof: Recall that if $u^\star$ is a possible steady-state value of u, then the corresponding steady-state value of $\hat{u} = G_{\text{con}}u$ is given by $\hat{u}^\star = Ju^\star$ or equivalently $\hat{u}_i^\star = \mathsf{ave}(u^\star)$. Suppose $(\xi_0^\star, v_0^\star, u^\star)$ is a steady-state value of (ξ_0, v_0, u) in (1). From (1a), we know $\xi_0^\star = A_0\xi_0^\star + B_0\,\mathsf{ave}(u^\star)$, which implies that (5a) is satisfied when $\xi_i^k = \xi_0^\star$ and $\hat{u}_i^k = \mathsf{ave}(u^\star)$ for all k. By checking (5b) and (5c) in a similar way, one can verify that $(\xi_0^\star, v_0^\star, u_i^\star, v_0^\star, \mathsf{ave}(u^\star))$ is a steady-value of $(\xi_i, v_i, u_i, \hat{v}_i, \hat{u}_i)$ in (5) for $i = 1, 2, \dots, n$. ∎

Theorem 3 holds similarly for distributed algorithms of the form (2). The only difference is that we no longer need to apply consensus tracking on v as in Theorem 3, because $\mathsf{ave}(u^k)$ is the only computation that prevents decentralization in (2).

Corollary 4. *Suppose G_{con} is a system that achieves consensus tracking, and $\xi^\star$ is a possible steady-state value of ξ in (2). Then, $\xi^\star$ is also a steady-state value of ξ in*

$$\begin{aligned}\xi_i^{k+1} &= A_i\xi_i^k + B_{\text{loc},i}u_i^k + B_i\hat{u}_i^k\\ v_i^k &= C_i\xi_i^k + D_{\text{loc},i}u_i^k + D_i\hat{u}_i^k\\ u_i^k &= \phi_i(v_i^k)\end{aligned}$$

for $i = 1, 2, \dots, n$, where $\hat{u} = G_{\text{con}}u$.

Theorem 3 and Corollary 4 give an equivalent decentralized algorithm that admits the same steady-state solution as the original algorithm. We give a few examples to illustrate how to use the result to decentralize an existing algorithm. We first give an example on the application to the gradient descent algorithm in Example 1.

Example 5 (Decentralized gradient descent)**.** We use the system in (3) as G_{con} and derive its state space model by defining $\zeta^k := \hat{s}^k - s^k$:

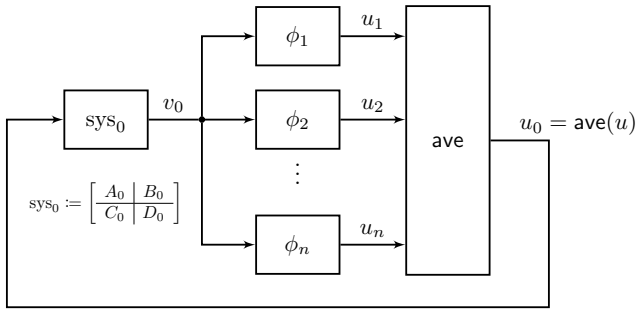
$$\zeta^{k+1} = W\zeta^k + (W - I)s^k, \qquad \hat{s}^k = \zeta^k + s^k, \qquad \zeta^0 = 0.$$

Then, we can apply Theorem 3 and obtain the following decentralized gradient descent algorithm:

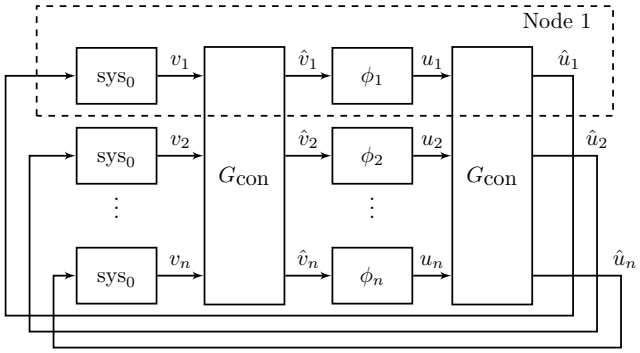
$$\xi^{k+1} = \xi^k - \eta\hat{u}^k, \qquad v^k = \xi^k, \qquad u^k = \nabla f(\hat{v}^k) \tag{6a}$$
$$\zeta_v^{k+1} = W\zeta_v^k + (W - I)v^k, \qquad \hat{v}^k = \zeta_v^k + v^k \tag{6b}$$
$$\zeta_u^{k+1} = W\zeta_u^k + (W - I)u^k, \qquad \hat{u}^k = \zeta_u^k + u^k, \tag{6c}$$

where the initial condition is given by $\zeta_v^0 = 0$ and $\zeta_u^0 = 0$. Equation (6a) retains the original dynamics of the (centralized) gradient descent algorithm. The new equations (6b) and (6c) are due to the consensus tracking $\hat{v} = G_{\text{con}}v$ and $\hat{u} = G_{\text{con}}u$ of v and u, respectively. It is not difficult to verify from (6) that $\hat{v}$ and u satisfy

$$\hat{v}^{k+2} = 2W\hat{v}^{k+1} - W^2\hat{v}^k - \eta(u^{k+1} - u^k),$$



(a) Centralized algorithm.



(b) Corresponding decentralized algorithm.

Fig. 1. Comparison between a centralized algorithm and the corresponding decentralized algorithm (Theorem 3) with consensus tracking.

which recovers the dynamics of the DIGing algorithm in [9]. This should not come as a surprise, because the DIGing algorithm is based on consensus tracking of the average gradient.

Next, we consider the ADMM algorithm in Example 2 and apply Corollary 4 to obtain a decentralized algorithm.

Example 6 (Decentralized ADMM)**.** We use the system given in (4) as G_{con} and apply Corollary 4 to the ADMM algorithm in Example 2. To make the algorithm more readable, we use the original optimization variables $x, y \in \mathbb{R}^{n\times d}$ instead of ξ and write the algorithm as

$$x^{k+1} = Jx^k - (y^k + w^k)/\rho, \qquad y^{k+1} = Jw^k - w^k$$
$$v^k = Jx^k - (y^k + w^k)/\rho, \qquad w^k = \nabla f(v^k).$$

To apply Corollary 4, we only need to replace Jx with the consensus tracking $\hat{x} = G_{\text{con}}x$ of x, and similarly for Ju. The corresponding decentralized algorithm is given by

$$x^{k+1} = \hat{x}^k - (y^k + w^k)/\rho, \qquad y^{k+1} = \hat{w}^k - w^k \tag{7a}$$
$$v^k = \hat{x}^k - (y^k + w^k)/\rho, \qquad w^k = \nabla f(v^k) \tag{7b}$$
$$\zeta_x^{k+1} = W\zeta_x^k + (W^2 - W)x^k, \qquad \hat{x}^k = \zeta_x^k + Wx^k \tag{7c}$$
$$\zeta_w^{k+1} = W\zeta_w^k + (W^2 - W)w^k, \qquad \hat{w}^k = \zeta_w^k + Ww^k, \tag{7d}$$

where the initial condition is given by $\zeta_x^0 = 0$ and $\zeta_w^0 = 0$. Similar to Example 5, equations (7a) and (7b) retain the original ADMM dynamics, except that Jx^k and Jw^k are replaced respectively by $\hat{x}^k$ and $\hat{w}^k$, which are the output from consensus tracking given in (7c) and (7d).

D. Discussions

The result in Theorem 3 can be viewed as a *separation principle* for designing decentralized optimization algorithms. Specifically, a decentralized optimization algorithm can be formed by a non-decentralized base optimization algorithm (e.g., gradient descent, ADMM) and a decentralized consensus tracking system $G_{\rm con}$. The system $G_{\rm con}$ can be viewed as an approximation of the averaging operator ave that appears in the base algorithm; the faster $G_{\rm con}$ reaches consensus, the better the approximation. The separation principle, however, does not require an explicit separation in time scale between the base algorithm and consensus tracking, which has been used in some previous work on decentralized gradient descent [2], [6].

For converting centralized algorithms of the form (1), an additional consensus tracking system $G_{\rm con}$ is required. The conversion procedure can be interpreted as follows. Recall that the computation in (1a)–(1b) still takes place centrally (within "$\rm sys_0$" in Fig. 1a) even when ave is replaced by a decentralized implementation. To make the computation decentralized, we create n identical copies of $\rm sys_0$, one at each node. Despite being identical to each other, each $\rm sys_0$ will generate a different output v_i, because we can no longer guarantee $\hat{u}_1 = \hat{u}_2 = \cdots = \hat{u}_n$ after ave is replaced by $G_{\rm con}$. Therefore, we need to use an additional $G_{\rm con}$ to enforce consensus among all copies of $\rm sys_0$.

To illustrate the importance of the additional $G_{\rm con}$ for centralized algorithms, consider again the gradient descent algorithm in Example 1. Without the additional $G_{\rm con}$, the resulting decentralized algorithm would become

$$\xi^{k+1} = \xi^k - \eta\hat{u}^k, \qquad v^k = \xi^k, \qquad u^k = \nabla f(v^k) \quad \text{(8a)}$$
$$\zeta_u^{k+1} = W\zeta_u^k + (W-I)u^k, \qquad \hat{u}^k = \zeta_u^k + u^k, \quad \text{(8b)}$$

where the initial condition is given by $\zeta_u^0 = 0$. In steady state, we must have $\hat{u}^\star = 0$ from (8a) and hence $\zeta_u^\star + u^\star = 0$ from (8b). We also know $\mathbf{1}^T\zeta_u^k = 0$ for all k based on the initial condition and (8b). From these, we can only conclude $\mathbf{1}^T u^\star = 0$ or equivalently $\sum_{i=1}^n \nabla f_i(v_i^\star) = 0$. Therefore, we cannot obtain an optimal solution unless we have $v_1^\star = v_2^\star = \cdots = v_n^\star$, which could have been enforced by the additional $G_{\rm con}$.

One benefit brought by the separation principle is that it allows us to derive a different decentralized algorithm by simply changing the consensus tracking system $G_{\rm con}$. For example, using the system in (4) instead as $G_{\rm con}$, we can obtain another decentralized gradient descent algorithm similar to (6), except that (6b) and (6c) are replaced by

$$\zeta_v^{k+1} = W\zeta_v^k + (W^2-W)v^k, \qquad \hat{v}^k = \zeta_v^k + Wv^k$$
$$\zeta_u^{k+1} = W\zeta_u^k + (W^2-W)u^k, \qquad \hat{u}^k = \zeta_u^k + Wu^k.$$

This can potentially permit the integration of existing results on dynamic average consensus [22] to handle other types of communication graphs (e.g., time-varying, directed). Another benefit of separation is reflected in the analysis of the resulting decentralized algorithm using the IQC framework proposed in [7], which provides automated convergence analysis of optimization algorithms. Separation allows immediate reuse of existing results derived for the base optimization algorithms, whereas it only remains to incorporate the consensus tracking system $G_{\rm con}$ into the IQC framework. This will be discussed in detail in Section III.

Compared with the recent result in [18] on unifying decentralized first-order optimization algorithms, we have not been able to recover certain gradient-based methods such as EXTRA [15]. Meanwhile, as shown in Example 6, our work is able to construct a new decentralized ADMM algorithm, whereas it is unclear whether the result in [18] is able to include decentralized first-order proximal methods (e.g., ADMM), which use gradients implicitly.

III. Convergence Analysis

We now show how to apply the IQC framework for automated convergence analysis of the decentralized algorithms obtained through Theorem 3. Throughout this section, we assume each f_i is μ-*strongly convex* and β-*smooth*, i.e., there exist $\mu > 0$ and $\beta > 0$ such that

$$\mu\left\|x-y\right\|^2 \le \left(\nabla f_i(x) - \nabla f_i(y)\right)^T(x-y) \le \beta\left\|x-y\right\|^2$$

holds for all $x, y \in \mathbb{R}^d$. This assumption enables us to simplify the presentation and is not a limitation of the IQC analysis framework. For example, a similar IQC-based analysis has been developed when the assumption on strong convexity is removed [3], [4], [5].

A. IQC preliminaries

Many optimization algorithms, including the gradient descent method and ADMM presented in Examples 1 and 2, can be viewed as a feedback interconnection of the form

$$\xi^{k+1} = A\xi^k + Bu^k, \qquad v^k = C\xi^k + Du^k \quad \text{(9a)}$$
$$u^k = \phi(v^k). \quad \text{(9b)}$$

We assume that the feedback interconnection is well-posed, which holds for both the gradient descent method and ADMM. In the view of (9), convergence analysis of an optimization algorithm becomes stability analysis of the interconnection (9). The IQC framework treats the nonlinearity ϕ as an uncertain system whose input v and output u are constrained by a quadratic inequality

$$(z^k - z^\star)^T M(z^k - z^\star) \ge 0 \quad \forall k, \quad (10)$$

where $z = \Psi(v,u)$ is a *filtered* version of (v,u):

$$\psi^{k+1} = A_\Psi\psi^k + B_\Psi\begin{bmatrix} v^k \\ u^k \end{bmatrix}, \quad z^k = C_\Psi\psi^k + D_\Psi\begin{bmatrix} v^k \\ u^k \end{bmatrix}.$$

(Equation (10) is a special form of IQC called *pointwise* IQC. Refer to [7] for more general IQCs.) For example, when $u^k = \phi(v^k) = \nabla f_0(v^k)$, based on strong convexity and smoothness of f_0, an IQC for ϕ is given by

$$(z^k - z^\star)^T\left(\begin{bmatrix} 2\mu\beta & \mu+\beta \\ * & -2 \end{bmatrix} \otimes I_d\right)(z^k - z^\star) \ge 0,$$

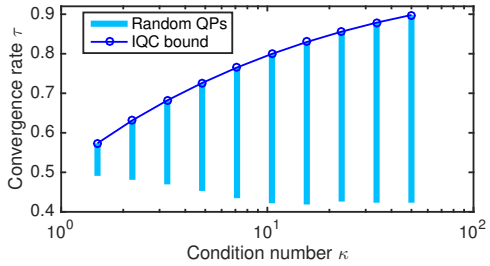
where $z^k = (v^k, u^k)$. It can be shown (cf. [7]) that (9) converges exponentially (linearly in the terminology used in optimization) with rate τ if there exists $P \succ 0$ such that

$$\begin{bmatrix} A^TPA - \tau^2 P & A^TPB \\ * & B^TPB \end{bmatrix} \otimes I_d + (*)^T (M \otimes I_d) \left(\begin{bmatrix} C & D \\ 0 & I \end{bmatrix} \otimes I_d \right) \preceq 0,$$

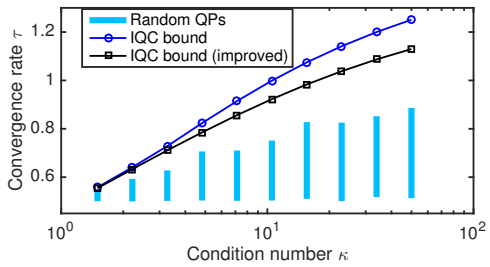
which is equivalent to

$$\begin{bmatrix} A^TPA - \tau^2 P & A^TPB \\ * & B^TPB \end{bmatrix} + [*]^T M \begin{bmatrix} C & D \\ 0 & I \end{bmatrix} \preceq 0.$$

The last step is called lossless dimensionality reduction [7, Sec. 4.2] and has a useful interpretation: for convergence analysis, we can assume $d = 1$ without loss of generality. This is consistent with the well-known fact that the convergence rate of many optimization algorithms does not depend on the dimension d of the optimization variable. From here on, we will assume $d = 1$ to simplify notation. For convenience, we will sometimes abuse the terminology and still use the term *convergence rate* even when $\tau > 1$, i.e., when the algorithm diverges.



(a) 2×2 gossip matrix W.



(b) 5×5 gossip matrix W.

Fig. 2. IQC analysis vs. simulation for randomly generated quadratic programs (10^5 runs, only maximum and minimum rates are shown). $\rho_0 = 1$.

B. Convergence analysis with a known gossip matrix

We now use the IQC framework to analyze the convergence of a decentralized algorithm obtained through Theorem 3. If the gossip matrix W in G_{con} is known, we can include the dynamics of G_{con} in (9a) while keeping the same IQC for the nonlinear map in (9b). For example, we can write the decentralized ADMM algorithm (7) in the form of (9) by choosing $\xi = (x, y, \zeta_x, \zeta_w)$. We adopt the normalization in [11] and choose $\rho = \rho_0\sqrt{\mu\beta}$ for some fixed $\rho_0 > 0$, which is equivalent to choosing $\rho = 1$, $\mu = \rho_0^{-1}\kappa^{-\frac{1}{2}}$, and $\beta = \rho_0^{-1}\kappa^{\frac{1}{2}}$, where $\kappa := \beta/\mu$ is the condition number. We did not optimize over ρ_0, which can be chosen by a line search. The IQCs are given by $(z^{j,k} - z^{j,\star})^T M_j (z^{j,k} - z^{j,\star}) \geq 0$ for $j = 1, 2, 3$, where $z^{1,k} = (v^k, w^k)$,

$$M_1 = \begin{bmatrix} 2\rho_0^{-2} I_n & \rho_0^{-1}(\kappa^{\frac{1}{2}} + \kappa^{-\frac{1}{2}}) I_n \\ * & -2I_n \end{bmatrix}, \qquad (11)$$

$z^{2,k} = \mathbf{1}^T \zeta_x^k$, $M_2 = -1$, $z^{3,k} = \mathbf{1}^T \zeta_w^k$, and $M_3 = -1$. The IQC (11) comes from the properties of f. The other two IQCs encode the constraints $\mathbf{1}^T \zeta_x^k = 0$ and $\mathbf{1}^T \zeta_w^k = 0$, which are a result of the dynamics given by (7c) and (7d) under the zero initial condition $\zeta_x^0 = 0$ and $\zeta_w^0 = 0$. We computed the convergence rate τ for

$$W = \begin{bmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{bmatrix}$$

and compare in Fig. 2a with simulated convergence rates for randomly generated quadratic programs. In this case, the bound on τ obtained from the IQC analysis is tight. The same IQC analysis is, however, not tight for an arbitrarily chosen 5×5 gossip matrix W, as can be seen from Fig. 2b. It is well known that the conservatism of IQC analysis can be reduced by enriching the class of IQCs used. For example, an improved upper bound (shown in black in Fig. 2b) can be obtained by replacing (11) with n similar IQCs for each (v_i, w_i) and adding the weighted off-by-one IQC in [7, Lem. 10]. The upper bound on τ exceeds 1 for large κ. It remains unclear whether this is because the upper bound is too conservative or because the algorithm actually diverges.

C. Convergence analysis with an unknown gossip matrix

The IQC analysis in Sec. III-B relies on knowing the exact gossip matrix W. If only the second-largest singular value $\sigma_2 := \sigma_{\max}(J - W)$ of W is known, which is a common assumption in the analysis of decentralized algorithms, it is no longer possible to include G_{con} directly into the system dynamics (9a). Instead, we choose to treat G_{con} as an uncertain system that can be characterized also by IQCs.

For the purpose of illustration, we will derive an IQC characterization of the consensus tracking system G_{con} given in (4), whose input is s and output is $\hat{s}$. Because the steady-state value of s and $\hat{s}$ are related by $\hat{s}^\star = Js^\star$, we define $\bar{s} := \hat{s} - Js$ so as to eliminate the steady-state component. Then, it can be shown that $\bar{s}$ and s satisfy

$$\bar{s}^{k+1} = W\bar{s}^k + (W - J)(s^{k+1} - s^k), \qquad \bar{s}^0 = (W - J)s^0.$$

As a result, we have $\mathbf{1}^T\bar{s}^0 = 0$ and $\mathbf{1}^T\bar{s}^{k+1} = \mathbf{1}^T\bar{s}^k$ for all k, i.e., $\mathbf{1}^T\bar{s}^k = 0$ for all k. We write $W = J + J_\perp \bar{W} J_\perp$, where $\bar{W} \in \mathbb{R}^{n \times n}$ satisfies $\sigma_{\max}(\bar{W}) = \sigma_2$. Then, we have

$$\bar{s}^{k+1} = J_\perp \bar{W} J_\perp (\bar{s}^k + s^{k+1} - s^k)$$

and hence

$$\left\|\bar{s}^{k+1}\right\| \leq \sigma_2 \left\|J_\perp(\bar{s}^k + s^{k+1} - s^k)\right\|, \qquad (12)$$

which can be described by an IQC with $z^{1,k} = (\bar{s}^k, J_\perp(\bar{s}^{k-1} + s^k - s^{k-1}))$ and

$$M_1 = \begin{bmatrix} -I & 0 \\ * & \sigma_2^2 I \end{bmatrix}.$$

We also add an IQC with $z^{2,k} = \bar{s}^k$ and $M_2 = -J$ to encode the constraint $\mathbf{1}^T \bar{s}^k = 0$, which was not captured by (12). Define a filter $\Psi_{\rm con}$ such that $(z^{1,\cdot}, z^{2,\cdot}) = \Psi_{\rm con}(s, \bar{s})$. We can carry out a convergence analysis of (7) by forming an interconnection of the following four systems: $(v, x) = G_{\rm ADMM}(w, \hat{x}, \hat{w})$, where $G_{\rm ADMM}$ is given by (7a)–(7b), $z_\nabla = (v, w)$, $z_x = \Psi_{\rm con}(x, \hat{x} - Jx)$, and $z_w = \Psi_{\rm con}(w, \hat{w} - Jw)$. The input to the interconnection is $(w, \hat{x}, \hat{w})$, and the output from the interconnection is (z_∇, z_x, z_w). The output obeys the following IQCs: z_∇ is constrained by M_1 in (11), and both z_x and z_w are constrained by the two matrices M_1 and M_2 associated with $\Psi_{\rm con}$. It can be verified that every matrix used in the IQC analysis is a block matrix whose blocks are linear combinations of J and $J_\perp$. Therefore, the dimensionality reduction in [17] can be applied so that the analysis does not depend on n.

We computed the worst-case convergence rate when only σ_2 is known. The result for different values of σ_2 is shown in Fig. 3. As expected, the convergence rate becomes slower as σ_2 increases. By comparing the result for $\sigma_2 = 0.2$ with Fig. 2a, it can be seen that the worst-case convergence analysis is more conservative. Similar to the IQC characterization (11) of the gradient, it is also possible to enrich the class of IQCs used for characterizing (12) to obtain a better upper bound on τ (cf. [19, Sec. 5.3], which was used in generating Fig. 3).

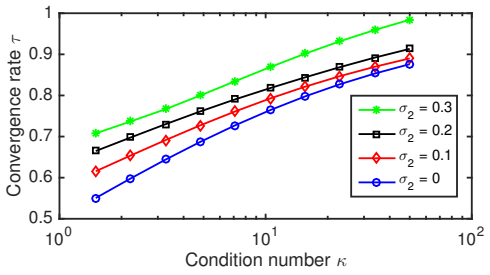


Fig. 3. IQC analysis based on the second-largest singular value σ_2 of W.

IV. Conclusions

We have proposed a separation principle for designing decentralized algorithms used in consensus optimization. Specifically, a decentralized optimization algorithm can be constructed by combining a non-decentralized base optimization algorithm and decentralized consensus tracking; the latter replaces the averaging operation that appears in the base algorithm. The separation principle provides modularity in both the design and analysis of algorithms. For design, the principle allows one to choose any combination of a base algorithm and a consensus tracking algorithm. For analysis, modularity is enabled by the automated convergence analysis based on IQCs, which is capable of integrating consensus tracking, regardless of whether the underlying gossip matrix is known. As a result, convergence of the decentralized algorithm can be readily analyzed as long as the base algorithm has an existing IQC characterization; the computation is as simple as calculating the interconnection of multiple linear dynamical systems coming from the base algorithm and consensus tracking. The workflow of design and analysis has been illustrated using a decentralized ADMM algorithm. We believe that the same principle also applies to other optimization problems that only require local information sharing, e.g., when locally coupled objective function and/or constraints are present (cf. [1, Sec. 7.2]).

References

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

[2] A. I. Chen and A. Ozdaglar. A fast distributed proximal-gradient method. In *Annual Allerton Conference on Communication, Control, and Computing*, pages 601–608, 2012.

[3] M. Fazlyab, A. Ribeiro, M. Morari, and V. M. Preciado. Analysis of optimization algorithms via integral quadratic constraints: Nonstrongly convex problems. *SIAM J. Optim.*, 28(3):2654–2689, 2018.

[4] S. Han. Computational convergence analysis of distributed gradient tracking for smooth convex optimization using dissipativity theory. *arXiv:1810.00257*, 2018.

[5] B. Hu and L. Lessard. Dissipativity theory for Nesterov's accelerated method. In *International Conference on Machine Learning (ICML)*, pages 1549–1557, 2017.

[6] D. Jakovetić, J. Xavier, and J. M. F. Moura. Fast distributed gradient methods. *IEEE Trans. Autom. Control*, 59(5):1131–1146, 2014.

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

[8] A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Trans. Autom. Control*, 42(6):819–830, 1997.

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

[10] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Autom. Control*, 54(1):48–61, 2009.

[11] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. A general analysis of the convergence of ADMM. In *International Conference on Machine Learning (ICML)*, pages 343–352, 2015.

[12] N. Parikh. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, 2014.

[13] G. Qu and N. Li. Accelerated distributed Nesterov gradient descent. *arXiv:1705.07176*, 2017.

[14] G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Trans. Control Netw. Syst.*, 5(3):1245–1260, 2018.

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

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

[17] A. Sundararajan, B. Hu, and L. Lessard. Robust convergence analysis of distributed optimization algorithms. In *Annual Allerton Conference on Communication, Control, and Computing*, 2017.

[18] A. Sundararajan, B. Van Scoy, and L. Lessard. A canonical form for first-order distributed optimization algorithms. *arXiv:1809.08709*, 2018.

[19] J. Veenman, C. W. Scherer, and H. Köroğlu. Robust stability and performance analysis based on integral quadratic constraints. *European Journal of Control*, 31:1–32, 2016.

[20] E. Wei and A. Ozdaglar. On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. *arXiv:1307.8254*, 2013.

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

[22] S. S. Kia, B. V. Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez. Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms. *IEEE Control Syst. Mag.*, 39(3):40–72, 2019.