Distributed Optimization Methods for Multi-Robot Systems

Part 2—A Survey

By Ola Shorinwa, Trevor Halsted, Javier Yu, and Mac Schwager

©SHUTTERSTOCK.COM/ANTONOV SERG

Although the field of distributed optimization is well developed, relevant literature focused on the application of distributed optimization to multi-robot problems is limited. This survey constitutes the second part of a two-part series on distributed optimization applied to multi-robot problems. In this article, we survey three main classes of distributed optimization algorithms—distributed first-order (DFO) methods, distributed sequential convex programming methods, and alternating direction method of multipliers (ADMM) methods—focusing on fully distributed methods that do not require coordination or computation by a central computer. We describe the fundamental structure of each category and note important variations around this structure, designed to address its associated drawbacks. Further, we provide practical implications of noteworthy assumptions made by distributed optimization algorithms, noting the classes of robotics problems suitable for these algorithms. Moreover, we identify important open research challenges in distributed optimization, specifically for robotics problem.

Digital Object Identifier 10.1109/MRA.2024.3352852
Date of publication 1 February 2024; date of current version 11 September 2024.

INTRODUCTION

In this article, we survey the literature in distributed optimization, specifically with an eye toward problems in multi-robot coordination. As we demonstrated in the first article in this two-part series [1], many multi-robot problems can be written as a sum of local objective functions, subject to an intersection of local constraints. Such problems can be solved with a powerful and growing arsenal of distributed optimization algorithms. Distributed optimization consists of multiple computation nodes working together to minimize a common objective function through local computation iterations and network-constrained communication steps, providing both computational and communication benefits by eliminating the need for data aggregation. Distributed optimization is also robust against the failure of individual nodes, as it does not

1070-9932/24©2024IEEE

rely on a central computation station, and many distributed optimization algorithms have inherent privacy-preserving properties, keeping the local data, objective function, and constraint function private to each robot while still allowing for all robots to benefit from one another. Distributed optimization has not yet been widely employed in robotics, and there exist many open opportunities for research in this space, which we highlight in this survey.

Although the field of distributed optimization is well established in many areas, such as computer networking and power systems, problems in robotics have a number of distinguishing features that are not often considered in the major application areas of distributed optimization. Notably, robots move, unlike their analogous counterparts in these other disciplines, which makes their networks time varying and prone to bandwidth limitations, packet drops, and delays. Robots often use optimization within a receding horizon or model predictive control (MPC) loop, so fast convergence to an optimal solution is essential in robotics. In addition, optimization problems in robotics are often constrained (e.g., with safety constraints, input constraints, or kinodynamics constraints in planning problems) and nonconvex [for example, simultaneous localization and mapping (SLAM) is a nonconvex optimization, as is trajectory planning and state estimation for any nonlinear robot model]. Many existing surveys on distributed optimization do not address these unique characteristics of robotics problems.

This survey constitutes the second part of a two-part series on distributed optimization for multi-robot systems. The first part consisted of a tutorial focused on the applicability of distributed optimization to multi-robot problems. In it, we demonstrated how a broad range of multi-robot problems can be cast in a form that is appropriate for distributed optimization, and we provided practical guidelines for implementing distributed optimization algorithms. In this survey, we highlight relevant distributed optimization algorithms and note the classes of robotics problems to which these algorithms can be applied. Noting the large body of work in distributed optimization, we categorize distributed optimization algorithms into three broad classes and identify the practical implications of these algorithms for robotics problems, including the challenges arising in the implementation of these algorithms on robotics platforms.

This survey is aimed at robotics researchers who are interested in research at the intersection of distributed optimization and multi-robot systems as well as robotics practitioners who want to harness the benefits of distributed optimization algorithms in solving practical robotics problems (see Figure 1). In this survey, we limit our discussion to optimization problems over real-valued decision variables. Although discrete optimization problems (i.e., integer programs or mixed-integer programs) arise in some robotics applications, these problems are beyond the scope of this survey. However, we note that distributed algorithms for integer and mixed-integer problems have been discussed in a number of different works [2], [3], [4]. Further, we limit our discussion to derivative-based methods, in contrast to derivative-free (zeroth-order) distributed optimization algorithms. We note that derivative-free optimization methods have been discussed extensively in [5], [6], [7], [8], [9], and [10].

In many robotics applications, such as field robotics, communication with a central computer (or the cloud) might be infeasible, even though each robot can communicate locally with other neighboring robots. Consequently, we focus particularly on distributed optimization algorithms that permit robots to use local robot-to-robot communication to compute an optimal solution, rather than algorithms that require coordination by a central computer. These methods yield a globally optimal solution for convex problems and, in general, a locally optimal solution for nonconvex problems, producing the same solution quality that would be obtained if a centralized method were applied. Although many distributed optimization algorithms are not inherently "online" (in the sense that these algorithms were not originally designed to be executed while the robot is actively gathering data or completing a task, providing information that changes its objective and constraint functions), we note that many of these algorithms can be applied in these online problems within the MPC framework, where a new optimization problem is solved periodically from streaming data.

In this survey, we provide a taxonomy of the different algorithms for performing distributed optimization, based on their defining mathematical characteristics. We identify three classes: DFO algorithms, distributed sequential convex programming, and distributed extensions to the ADMM:

1) *DFO algorithms*: The most common class of distributed optimization methods is based on the idea of averaging

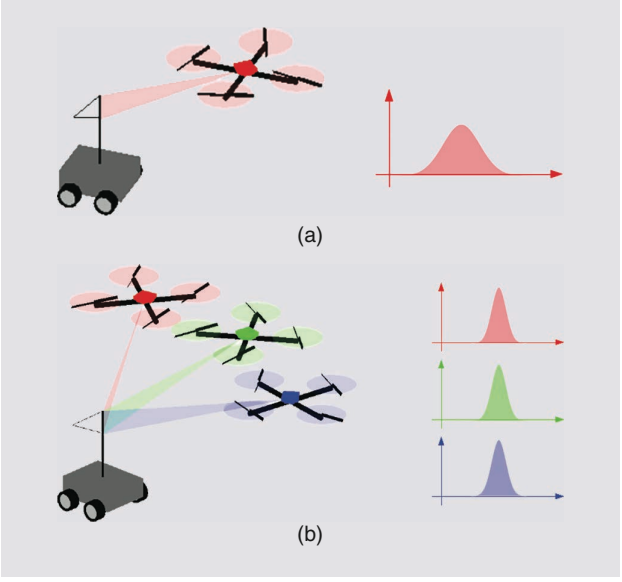

FIGURE 1. A motivation for distributed optimization: consider an estimation scenario in which a robot seeks to localize a target when given sensor measurements. (a) The robot can compute an optimal solution when given only its observations. (b) By using distributed optimization techniques, each robot in a networked system of robots can compute the optimal solution when given all robots' observations, without actually sharing individual sensor models or measurements with one another.

local gradients computed by each computational node to perform an approximate gradient descent update [11], and in this work, we refer to them as *DFO algorithms*. DFO algorithms can be further subdivided into distributed (sub)gradient descent, distributed gradient tracking, distributed stochastic gradient descent, and distributed dual averaging (DDA) algorithms, with each subcategory differing from the others based on the order of the update steps and the nature of the gradients used. In general, DFO algorithms use consensus methods to achieve a shared solution for the optimization problem. Many DFO algorithms allow for dynamic communication networks (including unidirectional and bidirectional networks) [12], [13] and limited computation resources [14], but they are often not well suited to constrained problems.

2) *Distributed sequential convex programming*: Sequential convex optimization is a common technique in centralized optimization that involves minimizing a sequence of convex approximations to the original (usually nonconvex) problem. Under certain conditions, the sequence of subproblems converges to a local optimum of the original problem. Newton's method and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method are common examples. The same concepts are used by a number of distributed optimization algorithms, and we refer to these algorithms as *distributed sequential convex programming methods*. Generally, these methods use consensus techniques to construct the convex approximations of the joint objective function. One example is the network Newton method [15], which uses consensus to approximate the inverse Hessian of the objective to construct a quadratic approximation of the joint problem. The NEXT family of algorithms [16] provides a flexible framework, which can utilize a variety of convex surrogate functions to approximate the joint problem and is specifically designed to optimize nonconvex objective functions. Although many distributed sequential convex programming methods are not suitable for problems with dynamic communication networks, a few distributed sequential convex programming algorithms are amenable to these problems [16].
3) *ADMM*: The last class of algorithms covered in this survey is based on the ADMM [17]. The ADMM works by minimizing the augmented Lagrangian of the optimization problem, using alternating updates to the primal and dual variables [18]. This method naturally accommodates constrained problems (with the assumption that we can convert inequality constraints to equality constraints by using slack variables). The original method is distributed but not in the sense we consider in this survey. Specifically, the original ADMM requires a central computation hub to collect all local primal computations from the nodes to perform a centralized dual update step. The ADMM was first modified to remove this requirement for a central node in [19], where it was used for distributed signal processing. The algorithm from [19] has since become known as the *consensus ADMM* (*C-ADMM*), although the original paper [19] did not use this terminology. A number of other distributed variants have been developed to address many unique characteristics, including unidirectional communication networks and limited communication bandwidth [20], [21], which are often present in robotics problems.

EXISTING SURVEYS

A number of other recent surveys on distributed optimization exist and provide useful background when working with the algorithms covered in this article. Some of these surveys cover applications of distributed optimization in distributed power systems [22], big data problems [23], and game theory [24], while others focus primarily on first-order methods for problems in multiagent control [25]. Other articles broadly address DFO optimization methods, including a discussion on the communication–computation tradeoffs [26], [27]. Another survey [28] covers exclusively nonconvex optimization in both batch and data streaming contexts but again analyzes only first-order methods. Finally, [29] covers a wide breadth of distributed optimization algorithms with a variety of assumptions, focusing exclusively on convex optimization problems. Building on the first article in this two-part series [1], which formulates multi-robot problems within the framework of distributed optimization, our survey differs from other existing surveys in that it specifically targets applications of distributed optimization to multi-robot problems: identifying suitable distributed optimization algorithms that address the practical issues arising in multi-robot problems and providing references demonstrating the application of distributed optimization to multi-robot problems. As a result, this survey highlights the practical implications of the assumptions made by many distributed optimization algorithms and provides a condensed taxonomic overview of useful methods for these applications. Other useful background material can be found for distributed computation [30], [31] and on multi-robot systems in [32] and [33].

CONTRIBUTIONS

This survey article has three primary objectives:

1) survey the literature across three different classes of distributed optimization algorithms, noting the defining mathematical characteristics of each category
2) highlight noteworthy assumptions made by distributed optimization algorithms and provide existing applications of distributed optimization algorithms to multi-robot problems
3) propose open research problems in distributed optimization for robotics.

ORGANIZATION

In the "Notation and Preliminaries" section, we introduce mathematical notation and preliminaries, and in the "Problem Formulation" section, we present the general formulation for the distributed optimization problem and describe the general framework shared by distributed optimization algorithms. The "DFO Algorithms," "Distributed Sequential

Convex Programming," and "ADMM" sections survey the literature in each of the three categories and provide details for representative algorithms in each category. The "Distributed Optimization in Robotics and Related Applications" section provides existing applications of distributed optimization in the robotics literature. In the "Research Opportunities in Distributed Optimization for Multi-robot Systems" section, we discuss open research problems in applying distributed optimization to multi-robot systems and robotics in general, and we offer concluding remarks in the "Conclusion" section.

NOTATION AND PRELIMINARIES

In this section, we introduce the notation used in this article and provide the definitions of mathematical concepts relevant to the discussion of the distribution optimization algorithms. We denote the gradient of a function $f:\mathbb{R}^n \to \mathbb{R}$ as ∇f and its Hessian as $\nabla^2 f$. We denote the vector containing all ones as $\mathbf{1}_n$, where n represents the number of elements in the vector. Next, we begin with the definition of stochastic matrices that arise in DFO optimization algorithms.

DEFINITION 1: NONNEGATIVE MATRIX

A matrix $W \in \mathbb{R}^{n\times n}$ is referred to as a *nonnegative matrix* if $w_{ij} \geq 0$ for all $i, j \in \{1, \ldots, n\}$.

DEFINITION 2: STOCHASTIC MATRIX

A nonnegative matrix $W \in \mathbb{R}^{n\times n}$ is referred to as a *row-stochastic matrix* if

$$W\mathbf{1}_n = \mathbf{1}_n. \tag{1}$$

In other words, the sum of all elements in each row of the matrix equals one. We refer to W as a *column-stochastic matrix* if

$$\mathbf{1}_n^\top W = \mathbf{1}_n^\top. \tag{2}$$

Likewise, for a doubly stochastic matrix W,

$$W\mathbf{1}_n = \mathbf{1}_n, \quad \text{and} \quad \mathbf{1}_n^\top W = \mathbf{1}_n^\top. \tag{3}$$

Now we provide the definition of some relevant properties of a sequence.

DEFINITION 3: SUMMABLE SEQUENCE

A sequence $\{\alpha(k)\}_{k\geq 0}$, with $k \in \mathbb{N}$, is a summable sequence if $\alpha(k) > 0$ for all k and

$$\sum_{k=0}^{\infty} \alpha(k) < \infty. \tag{4}$$

DEFINITION 4: SQUARE-SUMMABLE SEQUENCE

A sequence $\{\alpha(k)\}_{k\geq 0}$, with $k \in \mathbb{N}$, is a square-summable sequence if $\alpha(k) > 0$ for all k and

$$\sum_{k=0}^{\infty} (\alpha(k))^2 < \infty. \tag{5}$$

We next discuss some relevant notions of the connectivity of a graph.

DEFINITION 5: CONNECTIVITY OF AN UNDIRECTED GRAPH

An undirected graph $\mathcal{G}$ is connected if a path exists between every pair of vertices (i, j), where $i, j \in \mathcal{V}$. Note that such a path might traverse other vertices in $\mathcal{G}$.

DEFINITION 6: CONNECTIVITY OF A DIRECTED GRAPH

A directed graph $\mathcal{G}$ is strongly connected if a directed path exists between every pair of vertices (i, j), where $i, j \in \mathcal{V}$. In addition, a directed graph $\mathcal{G}$ is weakly connected if the underlying undirected graph is connected. The underlying undirected graph $\mathcal{G}_u$ of a directed graph $\mathcal{G}$ refers to a graph with the same set of vertices as $\mathcal{G}$ and a set of edges obtained by considering each edge in $\mathcal{G}$ a bidirectional edge. Consequently, every strongly connected directed graph is weakly connected; however, the converse is not true.

In distributed optimization in multi-robot systems, robots perform communication and computation steps to minimize some global objective function. We focus on problems in which the robots' exchange of information must respect the topology of an underlying distributed communication graph, which could possibly change over time. This communication graph, denoted as $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$, consists of vertices $\mathcal{V}(t) = \{1, \ldots, N\}$ and edges $\mathcal{E}(t) \subseteq \mathcal{V}(t) \times \mathcal{V}(t)$ over which pairwise communication can occur. For undirected graphs, we denote the set of neighbors of robot i as $\mathcal{N}_i(t)$. For directed graphs, we refer to the set of robots that can send information to robot i as the set of *in neighbors* of robot i, denoted by $\mathcal{N}_i^+(t)$. Likewise, for directed graphs, we refer to the set of robots that can receive information from robot i as the *out neighbors* of robot i, denoted by $\mathcal{N}_i^-(t)$.

DEFINITION 7: CONVERGENCE RATE

Provided that a sequence $\{x^{(k)}\}$ converges to $x^\star$, if there exists a positive scalar $r \in \mathbb{R}$, with $r \geq 1$, and a constant $\lambda \in \mathbb{R}$, with $\lambda > 0$, such that

$$\lim_{k\to\infty} \frac{\left\| x^{(k+1)} - x^\star \right\|}{\left\| x^{(k)} - x^\star \right\|^r} = \lambda \tag{6}$$

then r defines the order of convergence of the sequence $\{x^{(k)}\}$ to $x^\star$. Moreover, the asymptotic error constant is given by λ.

If $r = 1$ and $\lambda = 1$, then $\{x^{(k)}\}$ converges to $x^\star$ sublinearly. However, if $r = 1$ and $\lambda < 1$, then $\{x^{(k)}\}$ converges to $x^\star$ linearly. Likewise, $\{x^{(k)}\}$ converges to $x^\star$ quadratically if $r = 2$ and cubically if $r = 3$.

DEFINITION 8: SYNCHRONOUS ALGORITHM

An algorithm is synchronous if each robot (computational node) has to wait at a predetermined point for a specific message from other robots (computational nodes) before proceeding. In general, the end of an iteration of the algorithm represents the predetermined synchronization point. Conversely, in an

asynchronous algorithm, each robot completes each iteration at its own pace, without having to wait at a predetermined point. In other words, at any given time, the number of iterations of an asynchronous algorithm completed by each robot could differ from the number of iterations completed by other robots.

PROBLEM FORMULATION

We consider a general class of separable distributed optimization problems in which we express a joint objective function as the sum over local objective functions. From a multi-robot perspective, each robot knows only its own local function, but the robots collectively seek to find the optimum to the global function. In this general formulation, we also consider a set of joint constraints consisting of an intersection over local constraints. Each robot knows only its own local constraints and its local objective function. The resulting optimization problem is given by

$$\begin{aligned} \min_{x} \ & \sum_{i \in \mathcal{V}} f_i(x) \\ \text{subject to } & g_i(x) = 0 \quad \forall i \in \mathcal{V} \\ & h_i(x) \leq 0 \quad \forall i \in \mathcal{V} \end{aligned} \tag{7}$$

where $x \in \mathbb{R}^n$ denotes the optimization variable and $f_i : \mathbb{R}^n \to \mathbb{R}$, $g_i : \mathbb{R}^n \to \mathbb{R}$, and $h_i : \mathbb{R}^n \to \mathbb{R}$ denote the local objective function, equality constraint function, and inequality constraint function of robot i, respectively. The joint optimization problem (7) can be solved locally by each robot if all the robots share their objective and constraint functions with one another. Alternatively, the solution can be computed centrally if all the local functions are collated at a central station. However, robots typically possess limited computation and communication resources, which precludes each robot from sharing its local functions with other robots, particularly in problems with high-dimensional problem data, such as images, lidar, and other perception measurements.

Distributed optimization algorithms enable each robot to compute a solution of (7) locally without sharing its local objective, constraints, or data. These algorithms assign a copy of the optimization variable to each robot, enabling each robot to update its own copy locally and in parallel with the other robots. Moreover, distributed optimization algorithms enforce consensus among the robots for agreement on a common solution of the optimization problem. Consequently, these algorithms solve an equivalent reformulation of the optimization problem in (7), given by

$$\begin{aligned} \min_{\{x_i, \forall i \in \mathcal{V}\}} \ & \sum_{i \in \mathcal{V}} f_i(x_i) \\ \text{subject to } & x_i = x_j \quad \forall (i, j) \in \mathcal{E} \\ & g_i(x_i) = 0 \quad \forall i \in \mathcal{V} \\ & h_i(x_i) \leq 0 \quad \forall i \in \mathcal{V} \end{aligned} \tag{8}$$

where $x_i \in \mathbb{R}^n$ denotes robot i's local copy of the optimization variable. We note that the consensus constraints in (8) ensure agreement among all the robots, with the assumption that the communication graph is connected. Moreover, the consensus constraints are enforced between neighboring robots only, providing compatibility with a point-to-point communication network, where robots can communicate only with their one-hop neighbors. To simplify notation, we introduce the set $\mathcal{X}_i = \{x_i \,|\, g_i(x_i) = 0, h_i(x_i) \leq 0\}$, representing the feasible set given the constraint functions g_i and h_i. Consequently, we can express the problem in (8) succinctly, as follows:

$$\begin{aligned} \min_{\{x_i \in \mathcal{X}_i, \forall i \in \mathcal{V}\}} \ & \sum_{i \in \mathcal{V}} f_i(x_i) \\ \text{subject to } & x_i = x_j \quad \forall (i, j) \in \mathcal{E}. \end{aligned} \tag{9}$$

In the following sections, we discuss three broad classes of distributed optimization methods, namely, DFO methods, distributed sequential convex programming methods, and the ADMM. We note that DFO methods and distributed sequential convex programming methods implicitly enforce the consensus constraints in (9), while the ADMM enforces these constraints explicitly. While not all the methods that we survey explicitly address constraints of the form $g_i(x) = 0$, $h_i(x) \leq 0$, we note in each section considerations to accommodate these additional terms. In some cases, it is also appropriate to incorporate the constraints as penalty terms in the cost function.

Before proceeding, we highlight the general framework that distributed optimization algorithms share. Distributed optimization algorithms are iterative algorithms in which each robot executes a number of operations over discrete iterations $k = 0, 1, \ldots$ until convergence, where each iteration consists of a communication and computation step. During each communication round, each robot shares a set of its local variables with its neighbors, referred to as its "communicated" variables $\mathcal{Q}_i^{(k)}$, which we distinguish from its "internal" variables $\mathcal{P}_i^{(k)}$, which are not shared with its neighbors. In general, each algorithm requires initialization of the local variables of each robot in addition to algorithm-specific parameters, denoted by $\mathcal{R}_i^{(k)}$. We note that some algorithms require all the robots to utilize a common step-size at initialization; however, these parameters can be initialized prior to deployment of the robots.

DFO ALGORITHMS

The optimization problem in (7) (in its unconstrained form) can be solved through gradient descent, where the optimization variable is updated using

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}) \tag{10}$$

with $\nabla f(x^{(k)})$ denoting the gradient of the objective function at $x^{(k)}$, given by

$$\nabla f(x) = \sum_{i \in \mathcal{V}} \nabla f_i(x) \tag{11}$$

given some scheduled step-size $\alpha^{(k)}$. Inherently, computation of $\nabla f(x^{(k)})$ requires knowledge of the local objective functions or gradients by all robots in the network, which is infeasible in many problems.

DFO algorithms extend the centralized gradient scheme to the distributed setting, where robots communicate with one-hop neighbors without knowledge of the local objective functions

or gradients of all robots. In DFO methods, each robot updates its local variable by using a weighted combination of the local variables or gradients of its neighbors according to the weights specified by a stochastic weighting matrix W, allowing for the dispersion of information on the objective function or its gradient through the network. The stochastic matrix W must be compatible with the underlying communication network, with a nonzero element w_{ij} when robot j can send information to robot i.

From the perspective of a single robot, the update equations in DFO methods represent a tradeoff between the optimality of the robot's individual solution based on its local objective function and agreement with its neighbors. Consensus enables the robot to incorporate global information about the objective function's shape into its update, thereby allowing it to approximate a gradient descent step on the global cost function rather than on its local cost function.

Many DFO algorithms use a doubly stochastic matrix, a row-stochastic matrix [34], or a column-stochastic matrix, depending on the model of the communication network considered, while other methods use a push-sum approach. In addition, many methods further require symmetry of the doubly stochastic weighting matrix with $W = W^\top$. The weight matrix exerts a significant influence on the convergence rates of DFO algorithms, and thus, an appropriate choice of these weights is required for convergence of DFO methods.

The order of the update procedures for the local variables of each robot and the gradient used by each robot in performing its local update procedures differ among DFO algorithms, giving rise to four broad classes of DFO methods: distributed (sub)gradient descent and diffusion algorithms, gradient tracking algorithms, distributed stochastic gradient algorithms, and DDA. While distributed (sub)gradient descent algorithms require a decreasing step-size for convergence to an optimal solution, gradient tracking algorithms converge to an optimal solution without this condition. We discuss these distributed methods in the following sections.

DISTRIBUTED (SUB)GRADIENT DESCENT AND DIFFUSION ALGORITHMS

Tsitsiklis introduced a model for distributed gradient descent (DGD) in the 1980s in [35] and [11] (see also [30]). The works of Nedić and Ozdaglar in [14] revisit the problem, marking the beginning of interest in consensus-based frameworks for distributed optimization over the past decade. This basic model of DGD consists of an update term that involves consensus on the optimization variable as well as a step in the direction of the local gradient for each node:

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} - \alpha_i^{(k)} \nabla f_i\left(x_i^{(k)}\right) \tag{12}$$

where robot i updates its variable using a weighted combination of its neighbors' variables determined by the weights w_{ij}, with $\alpha_i(k)$ denoting its local step-size at iteration k.

For convergence to the optimal joint solution, these methods require the step-size to asymptotically decay to zero. As proved in [36], if $\alpha^{(k)}$ is chosen such that the sequence $\{\alpha^{(k)}\}$ is square summable but not summable, then the optimization variables of all robots converge to the optimal joint solution given the standard assumptions of a connected network, properly chosen weights, and bounded (sub)gradients. In contrast, the choice of a constant step-size for all time steps guarantees only the convergence of each robot's iterates to a neighborhood of the optimal joint solution. In practice, this means that a multi-robot system implementing DGD must coordinate on scheduling the decrease of the step-size. Nonetheless, DGD can generally tolerate some level of asynchrony or stochasticity. Algorithm 1 summarizes the update step for the DGD method in [14], with the step-size $\alpha^{(k)} = \alpha^{(0)}/\sqrt{k}$, with $\alpha^{(0)} > 0$.

We note that the update procedure given in (12) requires a doubly stochastic weighting matrix, which, in general, is incompatible with directed communication networks. Other DGD algorithms [37], [38], [39], [40] utilize the push-sum consensus protocol [41] in place of the consensus terms in (12), extending the application of DGD schemes to problems with directed communication networks.

In general, with a constant step-size, distributed (sub)gradient descent algorithms converge at a rate of $O(1/k)$ to a neighborhood of the optimal solution in convex problems [42]. With a decreasing step-size, some distributed (sub)gradient descent algorithms converge to an optimal solution at $O(\log k/k)$ by using an accelerated gradient scheme, such as the Nesterov gradient method [43].

DISTRIBUTED GRADIENT TRACKING ALGORITHMS

Although distributed (sub)gradient descent algorithms converge to an optimal joint solution, the requirement of a square-summable sequence $\{\alpha^{(k)}\}$, which results in a decaying step-size, reduces the convergence speed of these methods. Gradient tracking methods address this limitation by allowing each robot to utilize the changes in its local gradient between successive iterations as well as a local estimate of the average gradient across all robots in its update procedures, enabling the use of a constant step-size while retaining convergence to the optimal joint solution.

The EXTRA algorithm introduced by Shi et al. in [44] uses a fixed step-size while still achieving exact convergence.

ALGORITHM 1. DGD.

Initialization: $k \leftarrow 0, x_i^{(0)} \in \mathbb{R}^n$
Internal variables: $\mathcal{P}_i^{(k)} = \emptyset$
Communicated variables: $Q_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha^{(k)}, w_i)$
do in parallel $\forall i \in \mathcal{V}$
  Communicate $Q_i^{(k)}$ to all $j \in \mathcal{N}_i$
  Receive $Q_j^{(k)}$ from all $j \in \mathcal{N}_i$
  $\alpha^{(k)} = \dfrac{\alpha^{(0)}}{\sqrt{k}}$
  $x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} - \alpha^{(k)} \nabla f_i(x_i^{(k)})$
  $k \leftarrow k + 1$
while *stopping criterion is not satisfied*

EXTRA replaces the gradient term with the difference in the gradients of the previous two iterates. Because the contribution of this gradient difference term decays as the iterates converge to the optimal joint solution, EXTRA does not require the step-size to decay in order to converge to the exact optimal joint solution. EXTRA achieves linear convergence [42], and a variety of gradient tracking algorithms have since offered improvements on its linear rate [45] for convex problems with strongly convex objective functions.

The DIGing algorithm [46] whose update equations are shown in Algorithm 2, is one such similar method that extends the faster convergence properties of EXTRA to the domain of directed and time-varying graphs. DIGing requires communication of two variables, effectively doubling the communication cost per iteration when compared to DGD but greatly increasing the diversity of communication infrastructure that it can be deployed on.

Many other gradient tracking algorithms involve variations on the variables updated using consensus and the order of the update steps, such as NIDS [48] and Exact Diffusion [49], [50], [51], [52]. These algorithms, which generally require the use of doubly stochastic weighting matrices, have been extended to problems with row-stochastic or column-stochastic matrices [12], [13], [53], [54] and push-sum consensus [55] for distributed optimization in directed networks. To achieve faster convergence rates, many of these algorithms require each robot to communicate multiple local variables to its neighbors during each communication round. In addition, we note that some of these algorithms require all robots to use the same step-size, which can prove challenging in some situations. Several works offer a synthesis of various gradient tracking methods, noting the similarities among these methods. Under the canonical form proposed in [56] and [57], these algorithms and others differ only in the choice of several constant parameters. Jakoveti□ also provides a unified form for various gradient tracking algorithms in [58]. Some other works consider accelerated variants using Nesterov gradient descent [59], [60], [61]. Gradient tracking algorithms can be considered primal–dual methods with an appropriately defined augmented Lagrangian function [46], [62].

In general, gradient tracking algorithms address unconstrained distributed convex optimization problems, but these methods have been extended to nonconvex problems [63] and constrained problems using projected gradient descent [64], [65], [66]. Some other methods [67], [68], [69], [70] perform dual ascent on the dual problem of (7), where the robots compute their local primal variables from the related minimization problem by using their dual variables. These methods require doubly stochastic weighting matrices but allow for time-varying communication networks. DFO methods have been extended to the constrained setting [71], where each robot performs a subsequent proximal projection step to obtain solutions that satisfy the problem constraints.

In deep learning problems, the associated objective function often consists of a sum over a very large number of data points. Computing exact gradients for such problems can be prohibitively costly, so gradients are approximated by randomly sampling a subset of the data at each iteration and computing the gradient only over those data. Such methods, called *stochastic gradient descent*, dominate in deep learning. In [72], stochastic gradients are used in place of gradients in the DGD algorithm, and the resulting algorithm is shown to converge.

DDA

Dual averaging, first posed in [73] and extended in [74], takes a similar approach to distributed (sub)gradient descent methods in solving the optimization problem in (7), with the added benefit of providing a mechanism for handling problem constraints through a projection step, in a like manner to projected (sub) gradient descent methods. However, the original formulations of the dual averaging method require knowledge of all components of the objective function or its gradient, which is unavailable to all robots. The DDA method circumvents this limitation by modifying the update equations through a doubly stochastic weighting matrix to allow for updates of each robot's variable by using its local gradients and a weighted combination of the variables of its neighbors [75].

Similar to distributed (sub)gradient descent methods, DDA requires a sequence of decreasing step-sizes to converge to the optimal solution. Algorithm 3 provides the update equations

ALGORITHM 2. DIGing.

Initialization: $k \leftarrow 0$, $x_i^{(0)} \in \mathbb{R}^n$, $y_i^{(0)} = \nabla f_i(x_i^{(0)})$
Internal variables: $\mathcal{P}_i^{(k)} = \emptyset$
Communicated variables: $Q_i^{(k)} = (x_i^{(k)}, y_i^{k})$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha, w_i)$
do in parallel $\forall i \in \mathcal{V}$
  Communicate $Q_i^{(k)}$ to all $j \in \mathcal{N}_i$
  Receive $Q_j^{(k)}$ from all $j \in \mathcal{N}_i$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} - \alpha y_i^{(k)}$$
$$y_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} y_j^{(k)} + \nabla f_i(x_i^{(k+1)}) - \nabla f_i(x_i^{(k)})$$

  $k \leftarrow k + 1$
while *stopping criterion is not satisfied*

ALGORITHM 3. DDA.

Initialization: $k \leftarrow 0$, $x_i^{(0)} \in \mathbb{R}^n$, $z_i^{(0)} = x_i^{(0)}$
Internal variables: $\mathcal{P}_i = z_i^{(k)}$
Communicated variables: $Q_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha^{(k)}, w_i, \phi(\cdot))$
do in parallel $\forall i \in \mathcal{V}$
  Communicate $Q_i^{(k)}$ to all $j \in \mathcal{N}_i$
  Receive $Q_j^{(k)}$ from all $j \in \mathcal{N}_i$

$$z_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} z_j^{(k)} + \nabla f_i(x_i^{(k)})$$
$$x_i^{(k+1)} = \operatorname*{argmin}_{x \in \mathcal{X}_i} \left\{ x^\top z_i^{(k+1)} + \frac{1}{\alpha^{(k)}} \phi(x) \right\}$$

  $k \leftarrow k + 1$
while *stopping criterion is not satisfied*

in the DDA algorithm along with the projection step, which involves a proximal function $\phi(x)$, often defined as $(1/2)\|x\|_2^2$. After the projection step, the robot's variable satisfies the problem constraints described by the constraints set $\mathcal{X}$. Some of the same extensions made to distributed (sub)gradient descent algorithms have been studied for DDA, including analysis of the algorithm under communication time delays [76] and replacement of the doubly stochastic weighting matrix with push-sum consensus [77].

DISTRIBUTED SEQUENTIAL CONVEX PROGRAMMING

Sequential convex programming is a class of optimization methods, typically for nonconvex problems, that proceed iteratively by approximating the nonconvex problem with a convex surrogate computed from the current values of the decision variables. This convex surrogate is optimized, and the resulting decision variables are used to compute the convex surrogate for the next iterate. Newton's method is a classic example of a sequential convex method, in which the convex surrogate is a quadratic approximation of the original objective function. Several methods have been proposed for distributed sequential convex programming, as we survey here. As with DFO methods, distributed sequential convex programming takes the perspective of using consensus to approximate the global objective function, with the addition of approximating not only the global gradient but also the global Hessian. The benefit of this approach is that convergence typically requires fewer iterations and is less dependent on carefully selecting a step-size. This comes at the expense of requiring the robots to communicate more information in order to approximate the second-order characteristics of the global objective function.

APPROXIMATE NEWTON METHODS

Newton's method and its variants are commonly used for solving convex optimization problems, and they provide significant improvements in the convergence rate when second-order function information is available [78]. To apply Newton's method to the distributed optimization problem in (7), the network Newton-K (NN-K) algorithm [15] takes a penalty-based approach that introduces consensus among the robots' variables as components of the objective function. The NN-K method reformulates the constrained form of the distributed problem in (7) as the following unconstrained optimization problem:

$$\min_{\{x_i\in\mathbb{R}^n,\,\forall i\in\mathcal{V}\}} \alpha\sum_{i\in\mathcal{V}} f_i(x_i) + x_i^\top\left(\sum_{j\in\mathcal{N}\cup\{i\}}\bar{w}_{ij}x_j\right) \tag{13}$$

where $\bar{W} = I - W$ and α is a weighting hyperparameter.

However, the Newton descent step requires computing the inverse of the joint problem's Hessian, which cannot be directly computed in a distributed manner, as its inverse is dense. To allow for distributed computation of the Hessian inverse, the NN-K uses the first K terms of the Taylor series expansion $(I - X)^{-1} = \Sigma_{j=0}^{\infty} X^j$ to compute the approximate Hessian inverse, as introduced in [79]. Approximation of the Hessian inverse comes at an additional communication cost and requires an additional K communication rounds per update of the primal variable. Algorithm 4 summarizes the update procedures in the NN-K method, in which ϵ denotes the local step-size for the Newton step. Selection of the step-size parameter does not require any coordination among robots. As presented in Algorithm 4, the NN-K proceeds through two sets of update equations: an outer set of updates that initializes the Hessian approximation and computes the decision variable update and an inner Hessian approximation update; a communication round precedes the execution of either set of update equations. Increasing K, the number of intermediary communication rounds, improves the accuracy of the approximated Hessian inverse at the cost of increasing the communication cost per primal variable update.

A follow-up work optimizes a quadratic approximation of the augmented Lagrangian of the general distributed optimization problem (7) in which the primal variable update involves computing a P-approximate Hessian inverse to perform a Newton descent step, and the dual variable update uses gradient ascent [80]. The resulting exact second-order method (ESOM) algorithm provides a faster convergence rate than the NN-K at the cost of one additional round of communication for the dual ascent step. Notably, replacing the augmented Lagrangian in the ESOM formulation with its linear approximation results in the EXTRA update equations, showing the relationship between both approaches.

In some cases, computation of the Hessian is impossible because second-order information is not available or intractable due to the large dimensions of the problem. Quasi-Newton methods like the BFGS algorithm approximate the Hessian when it cannot be directly computed. The distributed BFGS (D-BFGS) algorithm [81] replaces the second-order information

ALGORITHM 4. NN-K.

Initialization: $k \leftarrow 0,\ x_i^{(0)} \in \mathbb{R}^n$
Internal variables: $\mathcal{P}_i^{(k)} = (g_i^{(k)}, D_i^{(k)})$
Communicated variables: $Q_i = (x_i^{(k)}, d_i^{(k+1)})$
Parameters: $\mathcal{R}_i = (\alpha, \epsilon, K, \bar{w}_i)$
do in parallel $\forall i \in \mathcal{V}$

$D_i^{(k+1)} = \alpha\nabla^2 f_i(x_i^{(k)}) + 2\bar{w}_{ii}I$

Communicate $x_i^{(k)}$ to all $j \in \mathcal{N}_i$

$g_i^{(k+1)} = \alpha\nabla f_i(x_i^{(k)}) + \sum_{j\in\mathcal{N}_i\cup\{i\}} \bar{w}_{ij}x_j^{(k)}$

$d_i^{(0)} = -(D_i^{(k+1)})^{-1} g_i^{(k+1)}$

for $p = 0$ **to** $K - 1$ **do**

Communicate $d_i^{(p)}$ to all $j \in \mathcal{N}_i$

$d_i^{(p+1)} = (D_i^{(k+1)})^{-1}\left[\bar{w}_{ii}d_i^{(p)} - g_i^{(k+1)} - \sum_{j\in\mathcal{N}_i\cup\{i\}} \bar{w}_{ij}d_j^{(p)}\right]$

end

$x_i^{(k+1)} = x_i^{(k)} + \epsilon\, d_i^{(K)}$

$k \leftarrow k + 1$

while *stopping criterion is not satisfied*

in the primal update in ESOM with a BFGS approximation (i.e., it replaces $D_i^{(k)}$ in a call to the Hessian approximation equations in Algorithm 4 with an approximation) and results in essentially a "doubly" approximate Hessian inverse. In [82], the D-BFGS method is extended so that the dual update also uses a distributed quasi-Newton update scheme rather than gradient ascent. The resulting primal–dual quasi-Newton method requires two consecutive iterative rounds of communication, doubling the communication overhead per primal variable update compared to its predecessors (NN-K, ESOM, and D-BFGS). However, the resulting algorithm is shown by the authors to still converge faster in terms of required communication. In addition, asynchronous variants of the approximate Newton methods have been developed [83].

CONVEX SURROGATE METHODS

While the approximate Newton methods in [80], [81], and [82] optimize a quadratic approximation of the augmented Lagrangian of (13), other distributed methods allow for more general and direct convex approximations of the distributed optimization problem. These convex approximations generally require the gradient of the joint objective function, which is inaccessible to any single robot. In the NEXT family of algorithms [16], dynamic consensus is used to allow each robot to approximate the global gradient, and that gradient is then used to compute a convex approximation of the joint objective function locally. A variety of surrogate functions $U(\cdot)$ are proposed, including linear, quadratic, and block convex functions, which allows for greater flexibility in tailoring the algorithm to individual applications. Using its surrogate of the joint objective function, each robot updates its local variables iteratively by solving its surrogate for the problem and then taking a weighted combination of the resulting solution with the solutions of its neighbors. To ensure convergence, NEXT algorithms require a series of decreasing step-sizes, resulting in generally slower convergence rates as well as additional hyperparameter tuning.

The SONATA [84] algorithm extends the surrogate function principles of NEXT and proposes a variety of nondoubly stochastic weighting schemes that can be used to perform gradient averaging similar to the push-sum protocols. The authors of SONATA also show that several configurations of the algorithm result in already proposed distributed optimization algorithms, including Aug-DGM [85], Push-DIG [46], and ADD-OPT [53].

ALGORITHM 5. NEXT.

Initialization: $k \leftarrow 0, x_i^{(0)} \in \mathbb{R}^n, y_i^{(0)} = \nabla f_i(x_i^{(0)}), \tilde{\pi}_i^{(k+1)} = N y_i^{(0)} - \nabla f_i(x_i^{(0)})$
Internal variables: $\mathcal{P}_i = (x_i^{(k)}, \tilde{x}_i^{(k)}, \tilde{\pi}_i^{(k)})$
Communicated variables: $Q_i^{(k)} = (z_i^{(k)}, y_i^{(k)})$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha^{(k)}, w_i, U(\cdot), \mathcal{X}_i)$
do in parallel $\forall i \in \mathcal{V}$

$$\tilde{x}_i^{(k)} = \operatorname*{argmin}_{x \in \mathcal{X}_i} U(x; x_i^{(k)}, \tilde{\pi}_i^{(k)})$$
$$z_i^{(k)} = x_i^{(k)} + \alpha^{(k)}(\tilde{x}_i^{(k)} - x_i^{(k)})$$

Communicate $Q_i^{(k)}$ to all $j \in \mathcal{N}_i$
Receive $Q_j^{(k)}$ from all $j \in \mathcal{N}_i$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} z_j^{(k)}$$
$$y_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} y_j^{(k)} + [\nabla f_i(x_i^{(k+1)}) - \nabla f_i(x_i^{(k)})]$$
$$\tilde{\pi}_i^{(k+1)} = N \cdot y_i^{(k+1)} - \nabla f_i(x_i^{(k+1)})$$
$$k \leftarrow k + 1$$

while *stopping criterion is not satisfied*

ADMM

Considering the optimization problem in (9) with only agreement constraints, we have

$$\min_{\{x_i \in \mathbb{R}^n, \forall i \in \mathcal{V}\}} \sum_{i \in \mathcal{V}} f_i(x_i) \tag{14}$$
$$\text{subject to } x_i = x_j \quad \forall (i,j) \in \mathcal{E}. \tag{15}$$

The method of multipliers solves this problem by alternating between minimizing the augmented Lagrangian of the optimization problem with respect to the primal variables $x_1, \ldots, x_n$ (the "primal update") and taking a gradient step to maximize the augmented Lagrangian with respect to the dual (the "dual update"). The augmented Lagrangian of (14) is given by

$$\mathcal{L}_a(\mathbf{x}, q) = \sum_{i=1}^{N} f_i(x_i) + \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \left(q_{i,j}^\top (x_i - x_j) + \frac{\rho}{2} \|x_i - x_j\|_2^2 \right) \tag{16}$$

where $q_{i,j}$ represents a dual variable for the consensus constraints between robots i and j, $q = [q_{i,j}^\top, \forall (i,j) \in \mathcal{E}]^\top$, and $\mathbf{x} = [x_1^\top, x_2^\top, \ldots, x_N^\top]^\top$. The parameter $\rho > 0$ represents a penalty term on the violations of the consensus constraints. The quadratic penalty term is what distinguishes the augmented Lagrangian, and it also distinguishes the method of multipliers from dual ascent. The main benefit of using the augmented Lagrangian is that the quadratic term essentially serves as a proximal operator and helps to ensure convergence.

In the ADMM, given the separability of the global objective function, the primal update is executed as successive minimizations over each primal variable (i.e., choose the minimizing x_1 with all other variables fixed, then choose the minimizing x_2, and so on). Most ADMM-based approaches do not satisfy our definition of *distributed* in that either the primal updates take place sequentially rather than in parallel or the dual update requires centralized computation [86], [87], [88]. However, the C-ADMM provides an ADMM-based optimization method that is fully distributed: the nodes alternate between updating their primal and dual variable and communicating with neighboring nodes [19], [89].

To achieve a distributed update of the primal and dual variables, the C-ADMM alters the agreement constraints among agents with an existing communication link by introducing auxiliary primal variables in (9) (instead of the constraint $x_i = x_j$, we have two constraints: $x_i = z_{ij}$ and $x_j = z_{ij}$). Considering the optimization steps across the entire network, the C-ADMM proceeds by optimizing the original primal variables, then the auxiliary primal variables, and then the dual variables, as in the

original formulation of the ADMM. We can perform minimization with respect to the primal variables and gradient ascent with respect to the dual variables on an augmented Lagrangian that is fully distributed among the robots. Further, we note that although the ADMM is typically applied to equality-constrained problems, the method can be extended to inequality-constrained problems quite easily. In particular, we note that inequality-constrained problems can be expressed as equality-constrained problems using indicator functions. With this approach, corresponding update procedures for constrained optimization problems can be derived using the ADMM.

Algorithm 6 summarizes the update procedures for the local primal and dual variables of each agent in constrained optimization problems, where y_i represents the dual variable that enforces agreement between robot i and its neighbors. We have incorporated the solution of the auxiliary primal variable update into the update procedure for $x_i^{(k+1)}$, noting that the auxiliary primal variable update can be performed implicitly $[z_{ij}^* = (1/2)(x_i + x_j)]$. The parameter ρ that weights the quadratic terms in $\mathcal{L}_a$ is also the step-size in the gradient ascent of the dual variable. We note that the update procedure for $x_i^{(k+1)}$ requires solving an optimization problem, which might be computationally intensive for certain objective functions. To simplify the update complexity, the optimization can be solved inexactly using a linear approximation of the objective function, such as [90], [91], and [92], or a quadratic approximation using the Hessian, such as decentralized quadratically approximated ADMM [93]. The convergence rate of ADMM methods depends on the value of the penalty parameter ρ. Several works discuss effective strategies for optimally selecting ρ [94]. In general, convergence of the C-ADMM and its variants is guaranteed only when the dual variables sum to zero, a condition that could be challenging to satisfy in problems with unreliable communication networks. Other distributed ADMM variants that do not require this condition have been developed [95], [96]. However, these methods incur a greater communication overhead to provide robustness in these problems. Gradient tracking algorithms are related to the C-ADMM when the minimization problem in the primal update procedure is solved using a single gradient decent update.

The C-ADMM, as presented in Algorithm 6, requires each robot to optimize over a local copy of the global decision variable x. However, many robotic problems have a fundamental structure that makes maintaining global knowledge at every individual robot unnecessary: each robot's data relate only to a subset of the global optimization variables, and each agent requires only a subset of the optimization variable for its role. For instance, in distributed SLAM, a memory-efficient solution would require a robot to optimize only over its local map and communicate with other robots only messages of shared interest. Other examples arise in distributed environmental monitoring by multiple robots [97]. The SOVA method [98] leverages the separability of the optimization variable to achieve orders-of-magnitude improvement in convergence rates, computation, and communication complexity over C-ADMM methods. The general approach of SOVA can also be found in partitioning-based methods, such as in [99], [100], and [101], which also accommodate asynchronous or lossy communication. Like SOVA, these methods exploit the partitioning of the state variables, in that robot i need not estimate the states that are not relevant to its local objective function.

In SOVA, each agent optimizes only over variables relevant to its data or role, enabling robotic applications in which agents have minimal access to computation and communication resources. SOVA introduces consistency constraints between each agent's local optimization variable and its neighbors, mapping the elements of the local optimization variables, given by

$$\Phi_{ij} x_i = \Phi_{ji} x_j \quad \forall j \in \mathcal{N}_i, \forall i \in \mathcal{V}$$

where Φ_{ij} and Φ_{ji} map elements of x_i and x_j to a common space. The C-ADMM represents a special case of SOVA where Φ_{ij} is always the identity matrix. The update procedures for each agent reduce to the equations given in Algorithm 7.

ALGORITHM 6. C-ADMM.

Initialization: $k \leftarrow 0$, $x_i^{(0)} \in \mathbb{R}^n$, $y_i^{(0)} = 0$
Internal variables: $\mathcal{P}_i^{(k)} = y_i^{(k)}$
Communicated variables: $\mathcal{Q}_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = \rho$
do in parallel $\forall i \in \mathcal{V}$

$$x_i^{(k+1)} = \operatorname*{argmin}_{x_i \in \mathcal{X}_i} \left\{ f_i(x_i) + x_i^\top y_i^{(k)} \cdots + \rho \sum_{j \in \mathcal{N}_i} \left\| x_i - \frac{1}{2}\left(x_i^{(k)} + x_j^{(k)}\right) \right\|_2^2 \right\}$$

Communicate $\mathcal{Q}_i^{(k)}$ to all $j \in \mathcal{N}_i$
Receive $\mathcal{Q}_j^{(k)}$ from all $j \in \mathcal{N}_i$

$$y_i^{(k+1)} = y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \left(x_i^{(k+1)} - x_j^{(k+1)} \right)$$

$k \leftarrow k + 1$
while *stopping criterion is not satisfied*

ALGORITHM 7. SOVA.

Initialization: $k \leftarrow 0$, $x_i^{(0)} \in \mathbb{R}^{n_i}$, $y_i^{(0)} = 0$
Internal variables: $\mathcal{P}_i^{(k)} = y_i^{(k)}$
Communicated variables: $\mathcal{Q}_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = \rho$
do in parallel $\forall i \in \mathcal{V}$

$$x_i^{(k+1)} = \operatorname*{argmin}_{x_i \in \mathcal{X}_i} \left\{ f_i(x_i) + x_i^\top y_i^{(k)} \cdots + \rho \sum_{j \in \mathcal{N}_i} \left\| \Phi_{ij} x_i - \frac{1}{2}\left(\Phi_{ij} x_i^{(k)} + \Phi_{ji} x_j^{(k)}\right) \right\|_2^2 \right\}$$

Communicate $\mathcal{Q}_i^{(k)}$ to all $j \in \mathcal{N}_i$
Receive $\mathcal{Q}_j^{(k)}$ from all $j \in \mathcal{N}_i$

$$y_i^{(k+1)} = y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \Phi_{ij}^\top \left(\Phi_{ij} x_i^{(k)} - \Phi_{ji} x_j^{(k)} \right)$$

$k \leftarrow k + 1$
while *stopping criterion is not satisfied*

DISTRIBUTED OPTIMIZATION IN ROBOTICS AND RELATED APPLICATIONS

In this section, we discuss some existing applications of distributed optimization to robotics problems. To simplify the presentation, we highlight a number of these applications in the following notable problems in robotics: synchronization, localization, mapping, and target tracking; online and deep learning problems; and task assignment, planning, and control. We refer the reader to the first article in this two-part series [1] for a case study on multidrone target tracking, which compares solutions using several different distributed optimization algorithms.

SYNCHRONIZATION, LOCALIZATION, MAPPING, AND TRACKING

Distributed optimization algorithms have found notable applications in robot localization from relative measurements [102], [103], including in networks with asynchronous communication [104]. More generally, DFO algorithms have been applied to optimization problems on manifolds, including $SE(3)$ localization [105], [106], [107], [108], synchronization problems [109], and formation control in $SO(3)$ [110], [111]. In pose graph optimization, distributed optimization has been employed through majorization–minimization schemes, which minimize an upper bound of the objective function [112], using gradient descent on Riemannian manifolds [113], [114], and block coordinate descent [115]. Other pose graph optimization methods have utilized distributed sequential programming algorithms using a quadratic approximation model of the nonconvex objective function, with Gauss–Seidel updates to enable distributed local computations among the robots [116]. Further, the ADMM has been employed in bundle adjustment and pose graph optimization problems, which involve the recovery of the 3D positions and orientations of a map and camera [117], [118], [119]. However, many of these algorithms require a central node for the dual variable updates, making them semidistributed. Nonetheless, a few fully distributed ADMM-based algorithms exist for bundle adjustment and cooperative localization problems [120], [121]. Other applications of distributed optimization arise in target tracking [122], signal estimation [19], and parameter estimation in global navigation satellite systems [123].

ONLINE AND DEEP LEARNING PROBLEMS

Distributed optimization has also been applied in online dynamic problems. In these problems, each robot gains knowledge of its time-varying objective function in an online fashion after taking an action or decision. A number of DFO algorithms have been designed for these problems [124], [125], [126]. Similarly, DDA has been adapted for online scenarios with both static communication graphs [127], [128] and time-varying communication topology [129], [130]. The push-sum variant of dual averaging has also been used for distributed training of deep learning algorithms and has been shown to be useful in avoiding pitfalls of other synchronous distributed training frameworks, which face notable challenges in problems with communication deadlocks [131]. Many of these algorithms emphasize parallelization.

In addition, distributed sequential convex programming algorithms have been developed for a number of learning problems where data are distributed, including semisupervised support vector machines [132], neural network training [133], and clustering [134]. Moreover, the ADMM has been applied to online problems, such as estimation and surveillance problems involving wireless sensor networks [135], [136]. The ADMM has also be applied to distributed deep learning in robot networks in [137].

TASK ASSIGNMENT, PLANNING, AND CONTROL

Distributed optimization has been applied to task assignment problems posed as optimization problems. Some works [138] employ distributed optimization using a distributed simplex method [139] to obtain an optimal assignment of the robots to a desired target formation. Other works employ the C-ADMM for distributed task assignment [140], [141]. Further applications of distributed optimization arise in motion planning [142], trajectory tracking problems involving teams of robots using nonlinear MPC [143], and collaborative manipulation [144], [145], which employs fully distributed variants of the ADMM. One feature common to these problems is that the joint decision variables, which consist of control inputs or action variables concatenated over all the robots, can often be partitioned so that each robot needs to consider only its own actions, as in [98], [99], [100], and [101]. This can lead to significantly faster convergence compared methods in which each agent has a complete copy of the joint decision variables, as discussed at the end of the "ADMM" section.

RESEARCH OPPORTUNITIES IN DISTRIBUTED OPTIMIZATION FOR MULTI-ROBOT SYSTEMS

In this section, we highlight challenges in the application of existing distributed optimization algorithms to multi-robot problems, each of which represents a promising direction for future research.

NONCONVEX AND CONSTRAINED ROBOTICS PROBLEMS

Distributed optimization methods have primarily focused on solving unconstrained convex optimization problems, which constitute a limited subset of robotics problems. Many robotics problems involve nonconvex objectives or constraints. For example, problems in multi-robot motion planning, SLAM, learning, distributed manipulation, and target tracking are often nonconvex and/or constrained.

Both DFO methods and C-ADMM methods can be modified for nonconvex and constrained problems; however, few examples of practical algorithms or rigorous analyses of performance for such modified algorithms exist in the literature. One way to implement the C-ADMM for nonconvex

problems is to solve each primal update step as a nonconvex optimization (e.g., through a quasi-Newton method or interior point method). Another option is to perform successive quadratic approximations in an outer loop and use the C-ADMM to solve each resulting quadratic problem in an inner loop. The tradeoff between these two options has not yet been explored in the literature, especially in the context of nonconvex problems in robotics.

BANDWIDTH-CONSTRAINED, LOSSY, OR DYNAMIC COMMUNICATION

In many robotics problems, each robot exchanges information with its neighbors over a communication network with a limited communication bandwidth, which effectively limits the size of the message packets that can be transmitted among robots. Moreover, in practical situations, the communication links among robots sometimes fail, resulting in packet losses. However, many distributed optimization methods do not consider communication among agents as an expensive unreliable resource, given that many of these methods were developed for problems with reliable communication infrastructure (e.g., multicore computing or computing in a hardwired cluster). Information quantization has been extensively employed in many disciplines to allow for the efficient exchange of information over bandwidth-constrained networks. Quantization involves encoding the data to be transmitted into a format that utilizes a lower number of bits, often resulting in lower precision. Transmission of the encoded data incurs a lower communication overhead, enabling each robot to communicate with its neighbors within the bandwidth constraints. A few distributed optimization algorithms have been designed for these problems, including quantized DFO algorithms. Some of these algorithms assume that all robots can communicate with a central node [146], [147], making them unsuitable for a variety of robotics of problems, while others do not make this assumption [148], [149], [150], [151]. In addition, quantized distributed variants of the ADMM also exist [21], [152], [153].

Generally, quantization introduces error between each robot's solution and the optimal solution. However, in some of these algorithms, the quantization error decays during the execution of the algorithms under certain assumptions on the quantizer and the quantization interval [148], [149]. However, quantization in distributed optimization algorithms generally results in slower convergence rates, which poses a challenge in robotics problems where a solution is required rapidly, such as MPC problems, highlighting the need for the development of more effective algorithms. Further, only a few distributed optimization algorithms consider problems with lossy communication networks [154], [155], [156].

In many practical situations, the communication network among robots changes as robots move, giving rise to a time-varying communication graph. While many DFO optimization algorithms [46] and some distributed sequential programming algorithms [16], [84] tolerate dynamic communication networks under the condition of bounded connectivity, in general, distributed ADMM algorithms are not amenable to problems with dynamic communication networks. This is an interesting avenue for future research.

LIMITED COMPUTATION RESOURCES

Another valuable direction for future research is in developing algorithms specifically for computationally limited robotic platforms, in which the timeliness of the solution is as important as the solution quality [157], [158]. In general, many distributed optimization methods involve computationally challenging procedures that require significant computational power, especially distributed methods for constrained problems [90], [91], [92]. These methods ignore the significance of computation time, assuming that agents have access to significant computational power. These assumptions often do not hold in robotics problems. Typically, robotics problems unfold over successive time periods, with an associated optimization phase at each step of the problem. Thus, agents must compute their solutions fast enough to proceed with computing a reasonable solution for the next problem, which requires efficient distributed optimization methods. Developing such algorithms specifically for multi-robot systems is an interesting topic for future work.

COORDINATION AND SYNCHRONIZATION

Many distributed optimization algorithms implicitly assume coordination in several aspects of implementation. First, while most algorithms accommodate an arbitrary initialization of the initial solution of each robot (at least in convex problems), they often place stringent requirements on the initialization of the algorithms' parameters. For instance, DFO methods assume a common step-size across all robots and in some cases a scheduled decrease in that step-size [14], [44], [46]. Similarly, DFO algorithms and distributed sequential convex programming algorithms require the specification of a stochastic matrix, which must be compatible with the underlying communication network. However, generating doubly stochastic matrices for directed communication networks is nontrivial if each robot does not know the global network topology [159]. The ADMM and its distributed variants require the selection of a common penalty parameter ρ.

Second, some DFO, distributed sequential programming, and distributed ADMM algorithms require synchronous execution (see Definition 8). If robots have variable computation times and a synchronous distributed optimization algorithm is being used, one solution is to implement a distributed barrier scheme, where each robot waits until all its neighbors have computed and communicated their most recent update before proceeding. However, barrier schemes can lead to significantly increased time to convergence, as some robots idle while waiting for their neighbors. To address this issue, a number of asynchronous distributed optimization algorithms have been developed [47], [81], [83], [121], [160], which allow each robot to perform its local updates asynchronously,

eliminating the need for synchronization. These asynchronous variants are guaranteed to converge to an optimal solution, provided that an integer $T \in \mathbb{Z}$ exists such that each robot performs at least one iteration of the algorithm over T time steps.

HARDWARE IMPLEMENTATION

Finally, we believe there is a gap between the analysis in the distributed optimization literature and the applicability of these distributed optimization algorithms to hardware implementations [26], [27], [29]. The suitability of algorithms to run efficiently and robustly on robots has still not be thoroughly proved. We provide empirical results of a hardware implementation of C-ADMM over XBee radios in the first article in this series [1]. While this survey considers adapting existing distributed optimization algorithms for robotic implementations, it could also be useful to consider the codesign of general-purpose distributed optimization algorithms with practical hardware setups.

CONCLUSION

Despite the amenability of many robotics problems to distributed optimization, few applications of distributed optimization to multi-robot problems exist. In this work, we have categorized distributed optimization methods into three broad classes—distributed first-order methods, distributed sequential convex programming methods, and the ADMM—highlighting the distinct mathematical techniques employed by these algorithms. Further, we have identified a number of important open challenges in distributed optimization for robotics, which could be interesting areas for future research. In general, the opportunities for research in distributed optimization for multi-robot systems are plentiful. Distributed optimization provides an appealing unifying framework from which to synthesize solutions for a large variety of problems in multi-robot systems.

ACKNOWLEDGMENT

This project was funded in part by National Science Foundation (NSF) National Robotics Initiative Awards 1830402 and 1925030. Trevor Halsted was supported by a National Defense Science and Engineering Graduate Fellowship, and Javier Yu was supported by an NSF Graduate Research Fellowship. Ola Shorinwa, Trevor Halsted, and Javier Yu contributed equally to this work.

AUTHORS

Ola Shorinwa, Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 USA. E-mail: shorinwa@stanford.edu.

Trevor Halsted, Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 USA. E-mail: halsted@stanford.edu.

Javier Yu, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA. E-mail: javieryu@stanford.edu.

Mac Schwager, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA. E-mail: schwager@stanford.edu.

REFERENCES

[1] O. Shorinwa, T. Halsted, J. Yu, and M. Schwager, "Distributed optimization methods for multi-robot systems: Part I—A tutorial," presented at the Amer. Control Conf. (ACC), 2023, pp. 1–8. [Online]. Available: https://msl.stanford.edu/papers/shorinwa_distributed_2023.pdf

[2] I. Prodan, F. Stoican, S. Olaru, C. Stoica, and S.-I. Niculescu, "Mixed-integer programming techniques in distributed MPC problems," in *Distributed Model Predictive Control Made Easy*, J. Maestre and R. Negenborn, Eds., Dordrecht, The Netherlands: Springer-Verlag, 2014, pp. 275–291.

[3] A. Murray, A. Engelmann, V. Hagenmeyer, and T. Faulwasser, "Hierarchical distributed mixed-integer optimization for reactive power dispatch," *IFAC-PapersOnLine*, vol. 51, no. 28, pp. 368–373, 2018, doi: 10.1016/j.ifacol.2018.11.730.

[4] A. Testa, A. Rucco, and G. Notarstefano, "Distributed mixed-integer linear programming via cut generation and constraint exchange," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1456–1467, Apr. 2020, doi: 10.1109/TAC.2019.2920812.

[5] S. Liu, P.-Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero III, and P. K. Varshney, "A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 43–54, Sep. 2020, doi: 10.1109/MSP.2020.3003837.

[6] D. Hajinezhad, M. Hong, and A. Garcia, "Zeroth order nonconvex multi-agent optimization over networks," 2017, *arXiv:1710.09997*.

[7] D. Hajinezhad, M. Hong, and A. Garcia, "ZONE: Zeroth-order nonconvex multiagent optimization over networks," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 3995–4010, Oct. 2019, doi: 10.1109/TAC.2019.2896025.

[8] D. Hajinezhad and M. Hong, "Perturbed proximal primal–dual algorithm for nonconvex nonsmooth optimization," *Math. Program.*, vol. 176, nos. 1–2, pp. 207–245, 2019, doi: 10.1007/s10107-019-01365-4.

[9] A. Beznosikov, E. Gorbunov, and A. Gasnikov, "Derivative-free method for composite optimization with applications to decentralized distributed optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 4038–4043, 2020, doi: 10.1016/j.ifacol.2020.12.2272.

[10] Y. Tang, J. Zhang, and N. Li, "Distributed zero-order algorithms for nonconvex multiagent optimization," *IEEE Trans. Control Netw. Syst.*, vol. 8, no. 1, pp. 269–281, Mar. 2021, doi: 10.1109/TCNS.2020.3024321.

[11] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986, doi: 10.1109/TAC.1986.1104412.

[12] F. Saadatniaki, R. Xin, and U. A. Khan, "Optimization over time-varying directed graphs with row and column-stochastic matrices," 2018, *arXiv:1810.07393*.

[13] R. Xin and U. A. Khan, "A linear algorithm for optimization over directed graphs with geometric convergence," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 315–320, Jul. 2018, doi: 10.1109/LCSYS.2018.2834316.

[14] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009, doi: 10.1109/TAC.2008.2009515.

[15] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network newton," in *Proc. 48th Asilomar Conf. Signals, Syst. Comput.*, 2014, pp. 1621–1625, doi: 10.1109/ACSSC.2014.7094740.

[16] P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 120–136, Jun. 2016, doi: 10.1109/TSIPN.2016.2524588.

[17] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM J. Control Optim.*, vol. 14, no. 5, pp. 877–898, 1976, doi: 10.1137/0314056.

[18] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011, doi: 10.1561/2200000016.

[19] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010, doi: 10.1109/TSP.2010.2055862.

[20] V. Khatana and M. V. Salapaka, "DC-DistADMM: ADMM algorithm for contrained distributed optimization over directed graphs," 2020, *arXiv:2003.13742*.

[21] S. Zhu, M. Hong, and B. Chen, "Quantized consensus ADMM for multi-agent distributed optimization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2016, pp. 4134–4138, doi: 10.1109/ICASSP.2016.7472455.

[22] D. K. Molzahn et al., "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Nov. 2017, doi: 10.1109/TSG.2017.2720471.

[23] G. Scutari and Y. Sun, "Parallel and distributed successive convex approximation methods for big-data optimization," in *Multi-Agent Optimization*, F. Facchinei and J. S. Pang, Eds., Cham, Switzerland: Springer-Verlag, 2018, pp. 141–308.

[24] B. Yang and M. Johansson, "Distributed optimization and games: A tutorial overview," *Networked Control Systems*, A. Bemporad, M. Heemels, and M. Johansson, Eds., London, U.K.: Springer-Verlag, pp. 109–148, 2010.

[25] A. Nedić and J. Liu, "Distributed optimization for control," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, no. 1, pp. 77–103, 2018, doi: 10.1146/annurev-control-060117-105131.

[26] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proc. IEEE*, vol. 106, no. 5, pp. 953–976, May 2018, doi: 10.1109/JPROC.2018.2817461.

[27] A. Nedić, "Distributed optimization over networks," in *Multi-Agent Optimization*, F. Facchinei and J. S. Pang, Eds., Cham, Switzerland: Springer-Verlag, 2018, pp. 1–84.

[28] T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang, and S. Lu, "Distributed learning in the nonconvex world: From batch data to streaming and beyond," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 26–38, May 2020, doi: 10.1109/MSP.2020.2970170.

[29] T. Yang et al., "A survey of distributed optimization," *Annu. Rev. Control*, vol. 47, pp. 278–305, Jun. 2019, doi: 10.1016/j.arcontrol.2019.05.006.

[30] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, vol. 23. Englewood Cliffs, NJ, USA: Prentice-Hall 1989.

[31] N. A. Lynch, *Distributed Algorithms*. New York, NY, USA: Elsevier, 1996.

[32] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks* (Applied Mathematics Series). Princeton, NJ, USA: Princeton Univ. Press, 2009. [Online]. Available: http://coordinationbook.info

[33] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*, vol. 33. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[34] V. S. Mai and E. H. Abed, "Distributed optimization over directed graphs with row stochasticity and constraint regularity," *Automatica*, vol. 102, pp. 94–104, Apr. 2019, doi: 10.1016/j.automatica.2018.07.020.

[35] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Cambridge Lab for Information and Decision Systems, Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep., 1984. [Online]. Available: https://www.mit.edu/~jnt/Papers/PhD-84-jnt.pdf

[36] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1291–1306, Jun. 2011, doi: 10.1109/TAC.2010.2091295.

[37] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 33–55, 2009, doi: 10.1137/060678324.

[38] A. Olshevsky, I. C. Paschalidis, and A. Spiridonoff, "Robust asynchronous stochastic gradient-push: Asymptotically optimal and network-independent performance for strongly convex functions," 2018, *arXiv:1811.03982*.

[39] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, Mar. 2015, doi: 10.1109/TAC.2014.2364096.

[40] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proc. IEEE Int. Symp. Inf. Theory*, 2010, pp. 1753–1757, doi: 10.1109/ISIT.2010.5513273.

[41] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. 44th Annu. IEEE Symp. Found. Comput. Sci.*, 2003, pp. 482–491, doi: 10.1109/SFCS.2003.1238221.

[42] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM J. Optim.*, vol. 26, no. 3, pp. 1835–1854, 2016, doi: 10.1137/130943170.

[43] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1131–1146, May 2014, doi: 10.1109/TAC.2014.2298712.

[44] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015, doi: 10.1137/14096668X.

[45] A. Daneshmand, G. Scutari, and V. Kungurtsev, "Second-order guarantees of distributed gradient algorithms," 2018, *arXiv:1809.08694*.

[46] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM J. Optim.*, vol. 27, no. 4, pp. 2597–2633, 2017, doi: 10.1137/16M1084316.

[47] S. Zheng et al., "Asynchronous stochastic gradient descent with delay compensation," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 4120–4129.

[48] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4494–4506, Sep. 2019, doi: 10.1109/TSP.2019.2926022.

[49] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—Part I: Algorithm development," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 708–723, Feb. 2019, doi: 10.1109/TSP.2018.2875898.

[50] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—Part II: Convergence analysis," *IEEE Trans. Signal Process.*, vol. 67, no. 3, pp. 724–739, Feb. 2019, doi: 10.1109/TSP.2018.2875883.

[51] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1245–1260, Sep. 2018, doi: 10.1109/TCNS.2017.2698261.

[52] R. Xin and U. A. Khan, "Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking," *IEEE Trans. Autom. Control*, vol. 65, no. 6, pp. 2627–2633, Jun. 2019, doi: 10.1109/TAC.2019.2942513.

[53] C. Xi, R. Xin, and U. A. Khan, "ADD-OPT: Accelerated distributed directed optimization," *IEEE Trans. Autom. Control*, vol. 63, no. 5, pp. 1329–1339, May 2018, doi: 10.1109/TAC.2017.2737582.

[54] C. Xi, V. S. Mai, R. Xin, E. H. Abed, and U. A. Khan, "Linear convergence in optimization over directed graphs with row-stochastic matrices," *IEEE Trans. Autom. Control*, vol. 63, no. 10, pp. 3558–3565, Oct. 2018, doi: 10.1109/TAC.2018.2797164.

[55] J. Zeng and W. Yin, "ExtraPush for convex smooth decentralized optimization over directed networks," *J. Comput. Math.*, vol. 35, no. 4, pp. 383–396, 2017, doi: 10.4208/jcm.1606-m2015-0452.

[56] A. Sundararajan, B. Van Scoy, and L. Lessard, "A canonical form for first-order distributed optimization algorithms," in *Proc. Amer. Control Conf.*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 4075–4080, doi: 10.23919/ACC.2019.8814838.

[57] A. Sundararajan, B. Van Scoy, and L. Lessard, "Analysis and design of first-order distributed optimization algorithms over time-varying graphs," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 4, pp. 1597–1608, Dec. 2020, doi: 10.1109/TCNS.2020.2988009.

[58] D. Jakovetić, "A unification and generalization of exact distributed first-order methods," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 1, pp. 31–46, Mar. 2019, doi: 10.1109/TSIPN.2018.2846183.

[59] G. Qu and N. Li, "Accelerated distributed Nesterov gradient descent," *IEEE Trans. Autom. Control*, vol. 65, no. 6, pp. 2566–2581, Jun. 2020, doi: 10.1109/TAC.2019.2937496.

[60] R. Xin, D. Jakovetić, and U. A. Khan, "Distributed Nesterov gradient methods over arbitrary graphs," *IEEE Signal Process. Lett.*, vol. 26, no. 8, pp. 1247–1251, Aug. 2019, doi: 10.1109/LSP.2019.2925537.

[61] Q. Lü, X. Liao, H. Li, and T. Huang, "A Nesterov-like gradient tracking algorithm for distributed optimization over directed networks," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 10, pp. 6258–6270, Oct. 2021, doi: 10.1109/TSMC.2019.2960770.

[62] F. Mansoori and E. Wei, "A general framework of exact primal-dual first-order algorithms for distributed optimization," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 6386–6391, doi: 10.1109/CDC40024.2019.9029902.

[63] T. Tatarenko and B. Touri, "Non-convex distributed optimization," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3744–3757, Aug. 2017, doi: 10.1109/TAC.2017.2648041.

[64] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *J. Optim. Theory Appl.*, vol. 147, no. 3, pp. 516–545, 2010, doi: 10.1007/s10957-010-9737-7.

[65] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE Trans. Autom. Control*, vol. 58, no. 2, pp. 391–405, Feb. 2013, doi: 10.1109/TAC.2012.2209984.

[66] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM J. Optim.*, vol. 20, no. 3, pp. 1157–1170, 2010, doi: 10.1137/08073038X.

[67] M. Maros and J. Jaldén, "PANDA: A dual linearly converging method for distributed optimization over time-varying undirected graphs," in *Proc. IEEE Conf. Decis. Control (CDC)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 6520–6525, doi: 10.1109/CDC.2018.8619626.

[68] M. Maros and J. Jaldén, "A geometrically converging dual method for distributed optimization over time-varying graphs," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2465–2479, Jun. 2021, doi: 10.1109/TAC.2020.3018743.

[69] M. Maros and J. Jaldén, "Eco-PANDA: A computationally economic, geometrically converging dual optimization method on time-varying undirected graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 5257–5261, doi: 10.1109/ICASSP.2019.8683797.

[70] K. Seaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, "Optimal algorithms for smooth and strongly convex distributed optimization in networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, JMLR, 2017, pp. 3027–3036.

[71] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," *Math. Program.*, vol. 180, nos. 1–2, pp. 237–284, 2020, doi: 10.1007/s10107-018-1355-4.

[72] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5336–5346.

[73] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Math. Program.*, vol. 120, no. 1, pp. 221–259, 2009, doi: 10.1007/s10107-007-0149-x.

[74] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *J. Mach. Learn. Res.*, vol. 11, pp. 2543–2596, Oct. 2010.

[75] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 592–606, Mar. 2012, doi: 10.1109/TAC.2011.2161027.

[76] K. I. Tsianos and M. G. Rabbat, "Distributed consensus and optimization under communication delays," in *Proc. 49th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Piscataway, NJ, USA: IEEE Press, 2011, pp. 974–982, doi: 10.1109/Allerton.2011.6120272.

[77] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *Proc. IEEE 51st IEEE Conf. Decis. Control (CDC)*, Piscataway, NJ, USA: IEEE Press, 2012, pp. 5453–5458, doi: 10.1109/CDC.2012.6426375.

[78] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[79] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network flow optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 905–920, Apr. 2014, doi: 10.1109/TAC.2013.2293221.

[80] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 507–522, Dec. 2016, doi: 10.1109/TSIPN.2016.2613678.

[81] M. Eisen, A. Mokhtari, A. Ribeiro, and A. We, "Decentralized quasi-Newton methods," *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2613–2628, May 2017, doi: 10.1109/TSP.2017.2666776.

[82] M. Eisen, A. Mokhtari, and A. Ribeiro, "A primal-dual quasi-Newton method for exact consensus optimization," *IEEE Trans. Signal Process.*, vol. 67, no. 23, pp. 5983–5997, Dec. 2019, doi: 10.1109/TSP.2019.2951216.

[83] F. Mansoori and E. Wei, "A fast distributed asynchronous newton-based optimization algorithm," *IEEE Trans. Autom. Control*, vol. 65, no. 7, pp. 2769–2784, Jul. 2020, doi: 10.1109/TAC.2019.2933607.

[84] Y. Sun and G. Scutari, "Distributed nonconvex optimization for sparse representation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Piscataway, NJ, USA: IEEE Press, 2017, pp. 4044–4048, doi: 10.1109/ICASSP.2017.7952916.

[85] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes," in *Proc. 54th IEEE Conf. Decis. Control (CDC)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 2055–2060, doi: 10.1109/CDC.2015.7402509.

[86] B. Houska, J. Frasch, and M. Diehl, "An augmented Lagrangian based algorithm for distributed nonconvex optimization," *SIAM J. Optim.*, vol. 26, no. 2, pp. 1101–1127, 2016, doi: 10.1137/140975991.

[87] N. Chatzipanagiotis, D. Dentcheva, and M. M. Zavlanos, "An augmented Lagrangian method for distributed optimization," *Math. Program.*, vol. 152, nos. 1–2, pp. 405–434, 2015, doi: 10.1007/s10107-014-0808-7.

[88] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *Proc. 52nd IEEE Conf. Decis. Control.*, Piscataway, NJ, USA: IEEE Press, 2013, pp. 3671–3676, doi: 10.1109/CDC.2013.6760448.

[89] H. Terelius, U. Topcu, and R. M. Murray, "Decentralized multi-agent optimization via dual decomposition," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 11245–11251, 2011, doi: 10.3182/20110828-6-IT-1002.01959.

[90] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 4051–4064, Aug. 2015, doi: 10.1109/TSP.2015.2436358.

[91] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 482–497, Jan. 2015, doi: 10.1109/TSP.2014.2367458.

[92] F. Farina, A. Garulli, A. Giannitrapani, and G. Notarstefano, "A distributed asynchronous method of multipliers for constrained nonconvex optimization," *Automatica*, vol. 103, pp. 243–253, May 2019, doi: 10.1016/j.automatica.2019.02.003.

[93] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "DQM: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 64, no. 19, pp. 5158–5173, Oct. 2016, doi: 10.1109/TSP.2016.2548989.

[94] A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson, "The ADMM algorithm for distributed quadratic problems: Parameter selection and constraint preconditioning," *IEEE Trans. Signal Process.*, vol. 64, no. 2, pp. 290–305, Jan. 2015, doi: 10.1109/TSP.2015.2480041.

[95] D. Meng, M. Fazel, and M. Mesbahi, "Proximal alternating direction method of multipliers for distributed optimization on weighted graphs," in *Proc. 54th IEEE Conf. Decis. Control (CDC)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 1396–1401, doi: 10.1109/CDC.2015.7402406.

[96] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 5082–5095, Oct. 2017, doi: 10.1109/TAC.2017.2677879.

[97] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "Distributed environmental monitoring with finite element robots," *IEEE Trans. Robot.*, vol. 36, no. 2, pp. 380–398, Apr. 2020, doi: 10.1109/TRO.2019.2936747.

[98] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *Proc. Amer. Control Conf. (ACC)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 3619–3626, doi: 10.23919/ACC45564.2020.9147590.

[99] T. Erseghe, "A distributed and scalable processing method based upon ADMM," *IEEE Signal Process. Lett.*, vol. 19, no. 9, pp. 563–566, Sep. 2012, doi: 10.1109/LSP.2012.2207719.

[100] N. Bastianello, R. Carli, L. Schenato, and M. Todescato, "A partition-based implementation of the relaxed ADMM for distributed convex optimization over lossy networks," in *Proc. IEEE Conf. Decis. Control (CDC)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 3379–3384, doi: 10.1109/CDC.2018.8619729.

[101] M. Todescato, N. Bof, G. Cavraro, R. Carli, and L. Schenato, "Partition-based multi-agent optimization in the presence of lossy and asynchronous communication," *Automatica*, vol. 111, Jan. 2020, Art. no. 108648, doi: 10.1016/j.automatica.2019.108648.

[102] V.-L. Dang, B.-S. Le, T.-T. Bui, H.-T. Huynh, and C.-K. Pham, "A decentralized localization scheme for swarm robotics based on coordinate geometry and distributed gradient descent," *MATEC Web Conf.*, vol. 54, Feb. 1–3, 2016, Art. no. 02002, doi: 10.1051/matecconf/20165402002.

[103] N. A. Alwan and A. S. Mahmood, "Distributed gradient descent localization in wireless sensor networks," *Arabian J. Sci. Eng.*, vol. 40, no. 3, pp. 893–899, 2015, doi: 10.1007/s13369-014-1552-2.

[104] M. Todescato, A. Carron, R. Carli, and L. Schenato, "Distributed localization from relative noisy measurements: A robust gradient based approach," in *Proc. Eur. Control Conf. (ECC)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 1914–1919, doi: 10.1109/ECC.2015.7330818.

[105] R. Tron and R. Vidal, "Distributed image-based 3-d localization of camera sensor networks," in *Proc. 48h IEEE Conf. Decis. Control (CDC) Held Jointly 2009 28th Chin. Control Conf.*, Piscataway, NJ, USA: IEEE Press, 2009, pp. 901–908, doi: 10.1109/CDC.2009.5400405.

[106] R. Tron and R. Vidal, "Distributed computer vision algorithms," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 32–45, May 2011, doi: 10.1109/MSP.2011.940399.

[107] R. Tron, *Distributed Optimization on Manifolds for Consensus Algorithms and Camera Network Localization*. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2012.

[108] R. Tron and R. Vidal, "Distributed 3-D localization of camera sensor networks from 2-D image measurements," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3325–3340, Dec. 2014, doi: 10.1109/TAC.2014.2351912.

[109] A. Sarlette and R. Sepulchre, "Consensus optimization on manifolds," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 56–76, 2009, doi: 10.1137/060673400.

[110] K.-K. Oh and H.-S. Ahn, "Formation control and network localization via orientation alignment," *IEEE Trans. Autom. Control*, vol. 59, no. 2, pp. 540–545, Feb. 2014, doi: 10.1109/TAC.2013.2272972.

[111] K.-K. Oh and H.-S. Ahn, "Distributed formation control based on orientation alignment and position estimation," *Int. J. Control Automat. Syst.*, vol. 16, no. 3, pp. 1112–1119, Jun. 2018, doi: 10.1007/s12555-017-0280-2.

[112] T. Fan and T. Murphey, "Majorization minimization methods for distributed pose graph optimization with convergence guarantees," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 5058–5065, doi: 10.1109/IROS45743.2020.9341063.

[113] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, "Asynchronous and parallel distributed pose graph optimization," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5819–5826, Oct. 2020, doi: 10.1109/LRA.2020.3010216.

[114] J. Knuth and P. Barooah, "Collaborative localization with heterogeneous inter-robot measurements by Riemannian optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, Piscataway, NJ, USA: IEEE Press, 2013, pp. 1534–1539, doi: 10.1109/ICRA.2013.6630774.

[115] Y. Tian, K. Khosoussi, and J. P. How, "Block-coordinate descent on the Riemannian staircase for certifiably correct distributed rotation and pose synchronization," 2019, *arXiv:1911.03721*.

[116] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *Int. J. Robot. Res.*, vol. 36, no. 12, pp. 1286–1311, 2017, doi: 10.1177/0278364917732640.

[117] R. Zhang, S. Zhu, T. Fang, and L. Quan, "Distributed very large scale bundle adjustment by global camera consensus," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 29–38.

[118] A. Eriksson, J. Bastian, T.-J. Chin, and M. Isaksson, "A consensus-based framework for distributed bundle adjustment," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1754–1762.

[119] S. Choudhary, L. Carlone, H. I. Christensen, and F. Dellaert, "Exactly sparse memory efficient SLAM using the multi-block alternating direction method of multipliers," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 1349–1356, doi: 10.1109/IROS.2015.7353543.

[120] K. Natesan Ramamurthy, C.-C. Lin, A. Aravkin, S. Pankanti, and R. Viguier, "Distributed bundle adjustment," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2017, pp. 2146–2154.

[121] S. Kumar, R. Jain, and K. Rajawat, "Asynchronous optimization over heterogeneous networks via consensus ADMM," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 1, pp. 114–129, Mar. 2017, doi: 10.1109/TSIPN.2016.2593896.

[122] O. Shorinwa, J. Yu, T. Halsted, A. Koufos, and M. Schwager, "Distributed multi-target tracking for autonomous vehicle fleets," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 3495–3501, doi: 10.1109/ICRA40945.2020.9197241.

[123] A. Khodabandeh and P. Teunissen, "Distributed least-squares estimation applied to GNSS networks," *Meas. Sci. Technol.*, vol. 30, no. 4, 2019, Art. no. 044005, doi: 10.1088/1361-6501/ab034e.

[124] K. Lu, G. Jing, and L. Wang, "Online distributed optimization with strongly pseudoconvex-sum cost functions," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 426–433, Jan. 2020, doi: 10.1109/TAC.2019.2915745.

[125] S. Shahrampour and A. Jadbabaie, "Distributed online optimization in dynamic environments using mirror descent," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 714–725, Mar. 2018, doi: 10.1109/TAC.2017.2743462.

[126] Y. Zhang, R. J. Ravier, M. M. Zavlanos, and V. Tarokh, "A distributed online convex optimization algorithm with improved dynamic regret," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Piscataway, NJ, USA: IEEE Press, 2019, pp. 2449–2454, doi: 10.1109/CDC40024.2019.9029474.

[127] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed optimization via dual averaging," in *Proc. 52nd IEEE Conf. Decis. Control*, Piscataway, NJ, USA: IEEE Press, 2013, pp. 1484–1489, doi: 10.1109/CDC.2013.6760092.

[128] S. Shahrampour and A. Jadbabaie, "Exponentially fast parameter estimation in networks using distributed dual averaging," in *Proc. 52nd IEEE Conf. Decis. Control*, Piscataway, NJ, USA: IEEE Press, 2013, pp. 6196–6201, doi: 10.1109/CDC.2013.6760868.

[129] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed convex optimization on dynamic networks," *IEEE Trans. Autom. Control*, vol. 61, no. 11, pp. 3545–3550, Nov. 2016, doi: 10.1109/TAC.2016.2525928.

[130] S. Lee, A. Nedić, and M. Raginsky, "Stochastic dual averaging for decentralized online optimization on time-varying communication graphs," *IEEE Trans. Autom. Control*, vol. 62, no. 12, pp. 6407–6414, Dec. 2017, doi: 10.1109/TAC.2017.2650563.

[131] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Piscataway, NJ, USA: IEEE Press, 2012, pp. 1543–1550, doi: 10.1109/Allerton.2012.6483403.

[132] S. Scardapane, R. Fierimonte, P. Di Lorenzo, M. Panella, and A. Uncini, "Distributed semi-supervised support vector machines," *Neural Netw.*, vol. 80, pp. 43–52, Aug. 2016, doi: 10.1016/j.neunet.2016.04.007.

[133] S. Scardapane and P. D. Lorenzo, "A framework for parallel and distributed training of neural networks," *Neural Netw.*, vol. 91, pp. 42–54, Jul. 2017, doi: 10.1016/j.neunet.2017.04.004.

[134] R. Altilio, P. Di Lorenzo, and M. Panella, "Distributed data clustering over networks," *Pattern Recognit.*, vol. 93, pp. 603–620, Sep. 2019, doi: 10.1016/j.patcog.2019.04.021.

[135] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1185–1197, Mar. 2014, doi: 10.1109/TSP.2013.2295055.

[136] H. F. Xu, Q. Ling, and A. Ribeiro, "Online learning over a decentralized network through ADMM," *J. Oper. Res. Soc. China*, vol. 3, no. 4, pp. 537–562, Dec. 2015, doi: 10.1007/s40305-015-0104-0.

[137] J. Yu, J. A. Vincent, and M. Schwager, "DiNNO: Distributed neural network optimization for multi-robot collaborative learning," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1896–1903, Apr. 2022, doi: 10.1109/LRA.2022.3142402.

[138] E. Montijano and A. R. Mosteo, "Efficient multi-robot formations using distributed optimization," in *Proc. 53rd IEEE Conf. Decis. Control*, Piscataway, NJ, USA: IEEE Press, 2014, pp. 6167–6172, doi: 10.1109/CDC.2014.7040355.

[139] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012, doi: 10.1016/j.automatica.2012.06.040.

[140] R. Haksar, O. Shorinwa, P. Washington, and M. Schwager, "Consensus-based ADMM for task assignment in multi-robot teams," in *Proc. Int. Symp. Robot. Res.*, 2019, pp. 35–51, doi: 10.1007/978-3-030-95459-8_3.

[141] O. Shorinwa, R. N. Haksar, P. Washington, and M. Schwager, "Distributed multi-robot task assignment via consensus ADMM," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1781–1800, Jun. 2023, doi: 10.1109/TRO.2022.3228132.

[142] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 521–529.

[143] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of multiple vessels via distributed nonlinear model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 2523–2528, doi: 10.23919/ECC.2018.8550178.

[144] O. Shorinwa and M. Schwager, "Scalable collaborative manipulation with distributed trajectory planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 1, Piscataway, NJ, USA: IEEE Press, 2020, pp. 9108–9115, doi: 10.1109/IROS45743.2020.9340957.

[145] O. Shorinwa and M. Schwager, "Distributed contact-implicit trajectory optimization for collaborative manipulation," in *Proc. Int. Symp. Multi-Robot Multi-Agent Syst. (MRS)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 56–65, doi: 10.1109/MRS50823.2021.9620665.

[146] F. Alimisis, P. Davies, and D. Alistarh, "Communication-efficient distributed optimization with quantized preconditioners," 2021, *arXiv:2102.07214*.

[147] Y. Yu, J. Wu, and L. Huang, "Double quantization for communication-efficient distributed optimization," in *Proc. Adv. Neural Inform. Process. Syst.*, 2019, vol. 32, pp. 4438–4449.

[148] Y. Pu, M. N. Zeilinger, and C. N. Jones, "Quantization design for distributed optimization," *IEEE Trans. Autom. Control*, vol. 62, no. 5, pp. 2107–2120, May 2016, doi: 10.1109/TAC.2016.2600597.

[149] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "An exact quantized decentralized gradient descent algorithm," *IEEE Trans. Signal Process.*, vol. 67, no. 19, pp. 4934–4947, Oct. 2019, doi: 10.1109/TSP.2019.2932876.

[150] C.-S. Lee, N. Michelusi, and G. Scutari, "Finite rate quantized distributed optimization with geometric convergence," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 1876–1880, doi: 10.1109/ACSSC.2018.8645345.

[151] H. Li, S. Liu, Y. C. Soh, and L. Xie, "Event-triggered communication and data rate constraint for distributed optimization of multiagent systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 11, pp. 1908–1919, Nov. 2017, doi: 10.1109/TSMC.2017.2694323.

[152] A. Elgabli, J. Park, A. S. Bedi, C. B. Issaid, M. Bennis, and V. Aggarwal, "Q-GADMM: Quantized group ADMM for communication efficient decentralized machine learning," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 164–181, Jan. 2021, doi: 10.1109/TCOMM.2020.3026398.

[153] S. Zhu and B. Chen, "Distributed average consensus with deterministic quantization: An ADMM approach," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 692–696, doi: 10.1109/GlobalSIP.2015.7418285.

[154] N. Bastianello, M. Todescato, R. Carli, and L. Schenato, "Distributed optimization over lossy networks via relaxed Peaceman-Rachford splitting: A robust ADMM approach," in *Proc. Eur. Control Conf. (ECC)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 477–482, doi: 10.23919/ECC.2018.8550322.

[155] N. Bastianello, R. Carli, L. Schenato, and M. Todescato, "Asynchronous distributed optimization over lossy networks via relaxed ADMM: Stability and linear convergence," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2620–2635, Jun. 2021, doi: 10.1109/TAC.2020.3011358.

[156] N. Bof, R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, "Multiagent Newton–Raphson optimization over lossy networks," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2983–2990, Jul. 2019, doi: 10.1109/TAC.2018.2874748.

[157] S. M. Trenkwalder, "Computational resources of miniature robots: Classification and implications," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2722–2729, Jul. 2019, doi: 10.1109/LRA.2019.2917395.

[158] M. Lahijanian et al., "Resource-performance tradeoff analysis for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1840–1847, Jul. 2018, doi: 10.1109/LRA.2018.2803814.

[159] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *Eur. J. Control*, vol. 18, no. 6, pp. 539–557, 2012, doi: 10.3166/EJC.18.539-557.

[160] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 3043–3052.