A Survey of Distributed Optimization Methods for Multi-Robot Systems

Trevor Halsted[1], Ola Shorinwa[1], Javier Yu[2], Mac Schwager[2]

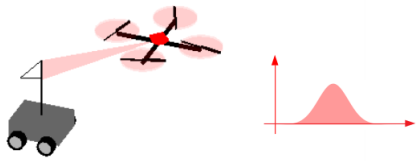
***Abstract*—Distributed optimization consists of multiple computation nodes working together to minimize a common objective function through local computation iterations and network-constrained communication steps. In the context of robotics, distributed optimization algorithms can enable multi-robot systems to accomplish tasks in the absence of centralized coordination. We present a general framework for applying distributed optimization as a module in a robotics pipeline. We survey several classes of distributed optimization algorithms and assess their practical suitability for multi-robot applications. We further compare the performance of different classes of algorithms in simulations for three prototypical multi-robot problem scenarios. The Consensus Alternating Direction Method of Multipliers (C-ADMM) emerges as a particularly attractive and versatile distributed optimization method for multi-robot systems.**

***Index Terms*—distributed optimization, multi-robot systems**

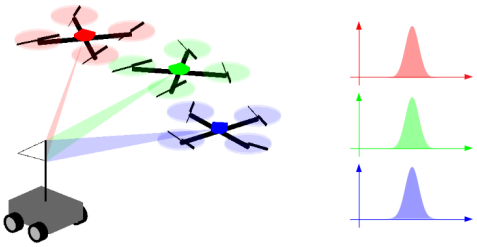
I. Introduction

Distributed optimization is the problem of minimizing a joint objective function consisting of a sum of several local objective functions, each corresponding to a computational node. While distributed optimization has been a longstanding topic of research in the optimization community (e.g., [1], [2]), its usage in robotics is limited to only a handful of examples. However, distributed optimization techniques have important implications for multi-robot systems, as we can cast many of the key tasks in this area, including cooperative estimation [3], multi-agent learning [4], and collaborative motion planning [5], as distributed optimization problems. The distributed optimization formulation offers a flexible and powerful paradigm for deriving efficient and distributed algorithms for many multi-robot problems. In this survey, we provide an overview of the distributed optimization literature, contextualize it for applications in robotics, discuss best practices for applying distributed optimization to robotic systems, and evaluate several algorithms in simulations for fundamental robotics problems.

We specifically consider optimization problems over real-valued decision variables (we do not consider discrete optimization, i.e., integer programs or mixed integer programs), and we assume that the robots communicate over a mesh network without central coordination. Often times, in the context of robotics, optimization is performed on-line within another iterative algorithm, e.g. for planning and control—as in Model Predictive Control (MPC), or perception—as in on-line Simultaneous Localization and Mapping (SLAM). Hence one can understand the algorithms in this paper as being run within a single time step of another iterative algorithm.



(a) Optimization by one robot yields the solution given only that robot's observations.



(b) Using distributed optimization, each robot obtains the optimal solution resulting from all robots' observations.

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

In evaluating the usefulness of distributed optimization algorithms for multi-robot applications, we focus on methods that permit robots to use local robot-to-robot communication to compute solutions that are of the same quality as those that would have been obtained were all the robots' observations available on one central computer. For convex problems, the robots all obtain a common globally optimal solution using distributed optimization. More generally, for non-convex problems, all robots obtain a common solution which may be locally optimal, but still of the same quality as that which would have been obtained by a single computer with all problem data.

In this survey, we provide a taxonomy of the range of different algorithms for performing distributed optimization based on their defining mathematical characteristics, and categorize the distributed optimization algorithms into three classes: distributed gradient descent, distributed sequential convex

*This project was funded in part by DARPA YFA award D18AP00064, and NSF NRI awards 1830402 and 1925030. The first author was supported on an NDSEG Fellowship, and the third author was supported on an NSF Graduate Research Fellowship.

** The first three authors contributed equally.

[1]Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA, {halsted, shorinwa}@stanford.edu

[2]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA {javieryu, schwager}@stanford.edu

programming, and distributed extensions to the alternating direction method of multipliers (ADMM). We do not discuss zeroth-order methods for distributed optimization [6], [7], [8].

Distributed Gradient Descent: The most common class of distributed optimization methods is based on the idea of averaging local gradients computed by each computational node to perform an approximate gradient descent [9], and in this work we refer to them as distributed gradient descent (DGD) methods. Accelerated DGD algorithms, like [10], offer significantly faster convergence rates over basic DGD. Furthermore, DGD methods have been shown to converge to the optimal solution on non-differentiable convex functions with subgradients [11] and with a push-sum averaging scheme [12], making them well suited for a broad range of applications.

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 non-convex) problem. Under certain conditions, the sequence of sub-problems 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 [13] 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 [14] provides a flexible framework which can utilize a variety of convex surrogate functions to approximate the joint problem, and is specifically designed to optimize non-convex objective functions.

Alternating Direction Method of Multipliers (ADMM): The last class of algorithms covered in this paper are based on the alternating direction method of multipliers (ADMM) [1]. ADMM works by minimizing the augmented Lagrangian of the optimization problem using alternating updates to the primal and dual variables [15]. The method is naturally amenable to constrained problems. 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. ADMM was first modified to remove this requirement for a central node in [16], where it was used for distributed signal processing. The algorithm from [16] has since become known as Consensus ADMM (C-ADMM), although that paper does not introduce this terminology. C-ADMM was later shown to have linear convergence rates on all strongly convex distributed optimization problems [17]. We find C-ADMM outperforms the other optimization algorithms that we tested in our three problem scenarios in terms of convergence speed, as well as computational and communication efficiency. In addition, we find that it is less sensitive to the choice of hyper-parameters than the other methods we tested. Therefore, C-ADMM emerges as an attractive option for problems in multi-robot systems.

Many problem features affect the convergence rates of each of these methods, and convergence guarantees are often qualified by the convexity of the underlying joint optimization problem. Essential to applying these algorithms is understanding their limitations and performance trade-offs. For instance, some methods are better suited for handling problems with constraints, but have a higher computational complexity or communication overhead.

Numerical results in this survey implement select algorithms from each of our taxonomic classes, and use them to solve three fundamental problems in multi-robot systems: cooperative target tracking, planning for coordinated package delivery, and cooperative multi-robot range-only mapping. Typically, distributed optimization algorithms are designed to solve convex problems, but often the optimization problems that arise in robotics applications are non-convex. The goal of these numerical simulations is to compare the performance of these algorithms not only in convex optimization where their convergence is often guaranteed, but also in non-convex problems where it is not.

A. Relevance to Robotics

While the field of distributed optimization is well-developed, its application to robotics problems remains nascent. As a result, many existing distributed optimization methods do not cater to the specific challenges that arise in robotics problems. Generally, robotics problems involve constrained non-convex optimization problems, an area not explored by many distributed optimization methods. Further, robots typically have limited access to significant computation and communication resources, placing greater importance on efficient optimization methods with low overhead. Many distributed optimization methods do not consider these issues with the assumption that agents have access to sufficient and reliable computation and communication infrastructure. For relevance to robotics problems, we specifically note methods that work on constrained problems and quantify the relative computation and communication costs incurred by distributed optimization methods.

In much of the literature, distributed optimization algorithms are compared on a convergence per iteration basis (per update to the decision variable). However, this scheme often obfuscates critical information for applications to robotics like local computation time, parameter sensitivity, and communication overhead. As part of the numerical results, this survey presents a more structured approach to analyzing the strengths and weaknesses of these algorithms in terms of metrics that matter for robotics research.

B. Existing Surveys

A number of other recent surveys on distributed optimization exist, and provide useful background when working with the algorithms covered in this survey. The survey [18] covers applications of distributed optimization for applications in distributed power systems, and [19] focuses on the application of predominately first-order methods to solving problems in multi-agent control. The article [20] broadly addresses

communication-computation trade-offs in distributed optimization, again focused mainly on first-order methods. Another survey [21], covers exclusively non-convex optimization in both batch and data-streaming contexts, but again only analyzes first-order methods. Finally, [22] covers a wide breadth of distributed optimization algorithms with a variety of assumptions focusing exclusively on convex optimization tasks. This survey differs from all of these in that it is specifically targeting optimization applications in robotics, and provides a condensed taxonomic overview of useful methods for these applications.

Other useful background material can be found for distributed computation [23] [24], and on multi-robot systems in [25] [26].

C. Contributions

This survey paper has four primary objectives:

1) Provide a unifying formulation for the general distributed optimization problem.
2) Develop a taxonomy for the different classes of distributed optimization algorithms.
3) Compare the performance of distributed optimization algorithms for applications in robotics with varying levels of difficulty.
4) Propose open research problems in distributed optimization for robotics.

D. Organization

In Section II, we present the general formulation for the distributed optimization problem, and give insight into the basic assumptions typically made while developing distributed optimization algorithms. Sections III - V provide greater detail on our algorithm classifications, and include details for representative algorithms. Finally, Section VII gives performance comparisons for select algorithms on a range of different optimization problems specifically highlighting communication versus computation trade-offs. In Sec. VIII we discuss open research problems in distributed optimization applied to multi-robot systems and robotics in general, and we offer concluding remarks in Sec. IX.

II. Problem Formulation

In distributed optimization in multi-robot systems, robots perform communication and computation steps to minimize some global cost function. We focus on problems in which the robots' exchange of information must respect the topology of an underlying distributed communication graph. This communication graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of vertices $\mathcal{V} = \{1, \dots, n\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ over which pairwise communication can occur. In general, we assume that the communication graph is connected (there exists some path of edges from any robot i to any other robot j) and undirected (if i can send information to j, then j can send information to i) but place no other assumptions on its structure.

In the general distributed optimization formulation, each robot $i \in \mathcal{V}$ can compute its local cost function $f_i(x)$ but cannot directly compute the local cost functions of the other robots. The robots' collective objective is to minimize the global cost function

$$f(x) = \sum_{i \in \mathcal{V}} f_i(x) \tag{1}$$

despite the limitations on the local information at each robot. The problem in (1) is separable in that it consists of the sum of the local cost functions.

In addition to local knowledge of the local objective functions, we assume that constraints on the optimization variable are only known locally as well. Therefore, the general form of the global cost function is

$$\begin{aligned} \min_{x} \ & \sum_{i \in \mathcal{V}} f_i(x) \\ \text{subject to } & g_i(x) = 0 \quad \forall i \in \mathcal{V} \\ & h_i(x) \leq 0 \quad \forall i \in \mathcal{V}. \end{aligned} \tag{2}$$

In this paper, we evaluate distributed algorithms on several classes of the separable optimization problem, including unconstrained linear least-squares, constrained convex, and constrained non-convex optimization problems.

Before describing the specific algorithms that solve distributed optimization problems, we first consider the general framework that all of these approaches share. Each algorithm progresses over discrete iterations $k = 0, 1, \dots$ until convergence. Besides assuming that each robot has the sole capability of evaluating its local cost function f_i, we also distinguish between the "private" variables $\mathcal{P}_i^{(k)}$ that the robot computes at each iteration k and the "public" variables $\mathcal{Q}_i^{(k)}$ that the robot communicates to its neighbors. Each algorithm also involves parameters $\mathcal{R}_i^{(k)}$, which generally require coordination among all of the robots but can typically be assigned before deployment of the system. Algorithm 1 describes the general framework for distributed optimization, in which each iteration k involves a communication step and a computation step.

In order for each robot to find the joint solution to (2) in a distributed manner, the update steps in algorithm 1 must allow the robots to cooperatively balance the costs accrued in each local cost function. From the perspective of a single robot, the update equations represent a trade-off between optimality of its individual solution based on its local cost function and agreement with its neighbors. In this paper, we distinguish

Algorithm 1 General Distributed Optimization Framework

function DistOpt($f_i, \mathcal{P}_i^{(0)}, \mathcal{Q}_i^{(0)}, \mathcal{R}_i^{(0)} \ \forall i \in \mathcal{V}$)
  $k \leftarrow 0$
  while stopping criterion is not satisfied **do**
    for $i \in \mathcal{V}$ **do** (in parallel)
      Communicate $\mathcal{Q}_i^{(k)}$ to all $j \in \mathcal{N}_i$
      Receive $\mathcal{Q}_j^{(k)}$ from all $j \in \mathcal{N}_i$
      Compute $\mathcal{P}_i^{(k+1)}, \mathcal{Q}_i^{(k+1)}, \mathcal{R}_i^{(k+1)}$
    end for
    $k \leftarrow k + 1$
  end while
end function

between two distinct perspectives on how this agreement is achieved.

In the following sections, we discuss three broad classes of distributed optimization methods, including distributed gradient descent, distributed sequential convex programming, and the alternating direction method of multipliers. In distributed gradient descent methods, the robots update their local variables using their local gradients and obtain the same solution of the global problem through consensus on their local variables. Distributed sequential convex programming methods take a similar approach but involve the problem Hessian, which provides information on the function curvature when updating the local variables.

The alternating direction method of multipliers takes a different approach by explicitly enforcing agreement between the robots' local variables before relaxing these constraints using the problem's Lagrangian. In this approach, we consider the reformulation of (1) to distinguish between each robots estimate of the decision variable x_i:

$$\begin{aligned} \min_{x_i \forall i \in \mathcal{V}} \; & \sum_{i \in \mathcal{V}} f_i(x_i) \\ \text{subject to } & x_i = x_j \quad \forall (i,j) \in \mathcal{E} \\ & g_i(x_i) = 0 \quad \forall i \in \mathcal{V} \\ & h_i(x_i) \leq 0 \quad \forall i \in \mathcal{V} \end{aligned} \tag{3}$$

where the agreement constraints are only enforced between neighboring robots. We discuss distributed gradient descent before proceeding with a discussion on distributed sequential convex programming and the alternating direction method of multipliers.

III. Distributed Gradient Descent

The optimization problem in (2) (in its unconstrained form) can be solved through gradient descent where the optimization variable is updated using

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}) \tag{4}$$

with $\nabla f(x^{(k)})$ denoting the gradient of the objective function, $\nabla f(x) = \sum_{i \in \mathcal{V}} \nabla f_i(x)$, 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.

Distributed Gradient Descent (DGD) methods extend the centralized gradient scheme to the distributed setting where robots communicate locally without necessarily having knowledge of the local objective functions or gradients of all robots. In DGD methods, each robot updates its local variable using a weighted combination of the local variables or gradients of its neighbors according to the weights specified by a weighting matrix W, allowing for the dispersion of information on the objective function or its gradient through the network. In general, the weighting matrix W reflects the topology of the communication network, with non-zero weights existing between pairs of neighboring robots. The weighting matrix exerts a significant influence on the convergence rates of DGD methods, and thus, an appropriate choice of these weights are required for convergence of DGD methods.

Algorithm 2 Distributed Gradient Descent (DGD)

Private variables: $\mathcal{P}_i = \emptyset$
Public variables: $\mathcal{Q}_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha^{(k)}, w_i)$
Update equations:

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

$$\alpha^{(k+1)} = \frac{\alpha^{(0)}}{\sqrt{k}}$$

Many DGD methods use a doubly stochastic matrix W, a row-stochastic matrix A [27], or a column-stochastic matrix B, 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 $W^\top \mathbf{1} = \mathbf{1}$ and $W = W^\top$.

The use of doubly stochastic matrices is supported by a rich body of work on the convergence of consensus algorithms of this form. (As observed in several works including [28], the standard average consensus problem is equivalent to using (5) to solve the distributed optimization problem (2) when the local cost functions f_i are constant).

DGD methods generally required decreasing step sizes for convergence to an optimal solution of the problem which negatively impacts their convergence rates; however, these methods have been modified to develop gradient descent methods with fixed step-sizes. Next, we discuss these distributed gradient descent variants.

A. Decreasing Step-Size DGD

Tsitsiklis introduced a model for DGD in the 1980s in [2] and [9] (see also [23]). The works of Nedić and Ozdaglar in [11] revisit the problem, marking the beginning of interest in consensus-based frameworks for distributed optimization over the recent 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:

$$x_i(k+1) = w_{ii} x_i(k) + \sum_{j \in \mathcal{N}_i} w_{ij} x_j(k) - \alpha_i(k) \nabla f_i(x_i(k)) \tag{5}$$

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 proven in [28], scheduling the step-size according to the rules $\sum_{k=1}^{\infty} \alpha_i(k) = \infty$ and $\sum_{k=1}^{\infty} \alpha_i(k)^2 < \infty$ with all robots' step-sizes equal guarantees the asymptotic convergence of the robots' optimization variables to the optimal joint solution, given the standard assumptions of a connected network, properly chosen weights, and bounded (sub)gradients. Alternatively, the choice of a constant step-size for all timesteps only guarantees convergence of each robot's iterates to a neighborhood of the optimal joint solution.

Algorithm 3 EXTRA

Private variables: $\mathcal{P}_i^{(k)} = \emptyset$
Public variables: $\mathcal{Q}_i^{(k)} = \left(x_i^{(k)}, x_i^{(k-1)}\right)$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha, w_i)$
Update equations:

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \left(x_j^{(k)} - \frac{1}{2} x_j^{(k-1)}\right) \cdots - \alpha \left[\nabla f_i\left(x_i^{(k)}\right) - \nabla f_i\left(x_i^{(k-1)}\right)\right]$$

Algorithm 4 Distributed Dual Averaging (DDA)

Private variables: $\mathcal{P}_i = z_i^{(k)}$
Public variables: $\mathcal{Q}_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = \left(\alpha^{(k)}, w_i, \phi(\cdot)\right)$
Update equations:

$$z_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} z_j^{(k)} + \nabla f_i\left(x_i^{(k)}\right)$$

$$x_i^{(k+1)} = \operatorname*{argmin}_{x \in \mathcal{X}} \left\{ x^\top z_i^{(k+1)} + \frac{1}{\alpha^{(k)}} \phi(x) \right\}$$

Algorithm 2 summarizes the update step for the decreasing step-size gradient descent method in Algorithm 1 with the step-size $\alpha^{(k+1)} = \frac{\alpha^{(0)}}{\sqrt{k}}$.

An alternative approach to the consensus term in (5) uses *push-sum* consensus introduced in the context of gossip-based distributed algorithms [29]. This approach involves two parallel consensus steps to circumvent the need for a doubly stochastic matrix which is replaced by a row-stochastic matrix A. The push-sum technique is also explored in [30], [31], [12], [32] in distributed gradient methods. In general, using a row-stochastic weighting matrix A in place of a doubly stochastic W would result in the consensus step becoming weighted according to the relative degrees of each robot in the communication graph. The push-sum approach introduces a second variable by which the robots keep track of their relative degrees and "unweight" the consensus term. The extra communication cost of this second variable comes with the benefit of extending consensus behavior to networks with asynchronous or directed communication.

Noting the sub-linear convergence rates of decreasing step-size DGD, some approaches have applied accelerated gradient schemes for improved convergence speed [33].

B. Fixed Step-Size DGD

Although decreasing step-size DGD methods converge to an optimal joint solution, the requirement of a decaying step-size reduces the convergence speed of these methods. Fixed step-size methods address this limitation by eliminating the need for decreasing step-sizes while retaining convergence to the optimal joint solution. The EXTRA algorithm introduced by Shi *et al.* in [10] uses a fixed step-size while still achieving exact convergence. 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 settle at the exact global solution. Algorithm 3 describes the update steps for the local variables of each robot in EXTRA. EXTRA achieves linear convergence [34], and a variety of DGD algorithms have since offered improvements on its linear rate [35]. Many other works with fixed step-sizes involve variations on the variables updated using consensus and the order of the update steps, including DIGing [36], [37], NIDS [38], Exact Diffusion [39], [40], and [41], [42]. These approaches which generally require the use of doubly stochastic weighting matrices have been extended to problems with row-stochastic or column-stochastic matrices [43], [44], [45], [46] and push-sum consensus [47] for distributed optimization in directed networks. Several works offer a synthesis of various fixed step-size DGD methods, noting the similarities between the fixed step-size DGD methods. Under the canonical form proposed in [48], these algorithms and others differ only in the choice of several constant parameters. We provide this unifying template for fixed step-size DGD methods in Algorithm 9, included in the Appendix X-A. Jakovetić also provides a unified form for various fixed-step-size DGD algorithms in [49]. Some other works consider accelerated variants using Nesterov gradient descent [50], [51], [50], [52].

In general, DGD methods address unconstrained distributed convex optimization problems, but these methods have been extended to non-convex problems [53] and constrained problems using projected gradient descent [54], [55], [56].

Some other methods perform dual-ascent on the dual problem of (2) [57], [58], [59], [60] where the robots compute their local primal variables from the related minimization problem using their dual variables. These methods require doubly stochastic weighting matrices but allow for time-varying communication networks. In [61], the robots perform a subsequent proximal projection step to obtain solutions which satisfy the problem constraints.

C. Distributed Dual Averaging

Dual averaging first posed in [62], and extended in [63], takes a similar approach to gradient descent methods in solving the optimization problem in (2), with the added benefit of providing a mechanism for handling problem constraints through a projection step in like manner as projected gradient descent methods. However, the original formulations of the dual averaging method requires knowledge of all components of the objective function or its gradient which is unavailable to all robots. The Distributed Dual Averaging method (DDA) circumvents this limitation by modifying the update equations using a doubly stochastic weighting matrix to allow for updates of each robot's variable using its local gradients and a weighted combination of the variables of its neighbors [64].

Similar to decreasing step-size DGD methods, distributed dual averaging requires a sequence of decreasing step-sizes to converge to the optimal solution. Algorithm 4 provides the

update equations in the DDA method, along with the projection step which involves a proximal function $\psi(x)$, often defined as $\frac{1}{2}\|x\|_2^2$. After the projection step, the robot's variable satisfy the problem constraints described by the constraints set $\mathcal{X}$.

Some of the same extensions made to DGD have been studied for DDA including analysis of the algorithm under communication time delays [65], and replacement of the doubly stochastic weighting matrix with push-sum consensus [66].

Applications of DGD

Distributed gradient descent methods have found notable applications in robot localization from relative measurements problems [67], [68] including in networks with asynchronous communication [69]. DGD has also been applied to optimization problems on manifolds including $SE(3)$ localization [70], [71], [72], [73], synchronization problems [74], and formation control in $SO(3)$ [75], [76]. Other works [77] employ DGD along with a distributed simplex method [78] to obtain an optimal assignment of the robots to a desired target formation. In pose graph optimization, DGD has been employed through majorization minimization schemes which minimize an upper-bound of the objective function [79] and using gradient descent on Riemannian manifolds [80], [81], and [82] (block-coordinate descent).

Online problems are a field of particular interest for distributed optimization algorithms, and a number of works adapted DDA for online scenarios, [83], [84], with several implemented in scenarios with time varying communication topology [85], [86]. 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 distributed training frameworks including communication deadlocks and asynchronous update steps [87].

IV. Distributed Sequential Convex Programming

A. Approximate Newton Methods

Newton's method, and its variants, are commonly used for solving convex optimization problems, and provide significant improvements in convergence rate when second-order function information is available [88]. While the distributed gradient descent methods exploit only information on the gradients of the objective function, Newton's method uses the Hessian of the objective function, providing additional information on the function's curvature which can improve convergence. To apply Newton's method to the distributed optimization problem in (2), the Network Newton-K (NN-K) algorithm [13] takes a penalty-based approach which introduces consensus between the robots' variables as components of the objective function. The NN-K method reformulates the constrained form of the distributed problem in (2) as the following unconstrained optimization problem:

$$\min_{x\forall i\in\mathcal{V}} \ \alpha\sum_{i\in\mathcal{V}} f_i(x_i) + x_i^\top\Big(\sum_{j\in\mathcal{N}\cup\{i\}} \bar{w}_{ij}x_j\Big) \tag{6}$$

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

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, NN-K uses the first K terms of the Taylor series expansion $(I-X)^{-1} = \sum_{j=0}^{\infty} X^j$ to compute the approximate Hessian inverse, as introduced in [89]. Approximation of the Hessian inverse comes at an additional communication cost, and requires an additional K communication rounds per update of the primal variable. Algorithms 5 summarizes the update procedures in the NN-K method in which ϵ denotes the step-size for the Newton's step. As presented in Algorithm 5, 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.

Algorithm 5 Network Newton-K (NN-K)

Private variables: $\mathcal{P}_i^{(k)} = \Big(g_i^{(k)}, D_i^{(k)}\Big)$
Public variables: $\mathcal{Q}_i = \Big(x_i^{(k)}, d_i^{(k+1)}\Big)$
Parameters: $\mathcal{R}_i = (\alpha, \epsilon, K, \bar{w}_i)$
Outer update equations:

$$D_i^{(k+1)} = \alpha\nabla^2 f_i(x_i^{(k)}) + 2\bar{w}_{ii}I$$
$$g_i^{(k+1)} = \alpha\nabla f_i(^{(k)}) + \sum_{j\in\mathcal{N}_i\cup\{i\}} \bar{w}_{ij}x_j^{(k)}$$
$$d_i^{(0)} = -\Big(D_i^{(k+1)}\Big)^{-1} g_i$$
$$\triangleright \textit{ compute } d_i^{(k+1)} \textit{ via } K \textit{ inner updates}$$
$$x_i^{(k+1)} = x_i^{(k)} + \epsilon\, d_i^{(k+1)}$$

Inner update equations: (Hessian approximation)

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

A follow-up work optimizes a quadratic approximation of the augmented Lagrangian of the general distributed optimization problem (2) where 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 [90]. The resulting algorithm Exact Second Order Method (ESOM) provides a faster convergence rate than 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. Quasi-Newton methods like the Broyden-Flectcher-Goldman-Shanno (BFGS) algorithm approximate the Hessian when it cannot be directly computed. The distributed BFGS (D-BFGS) algorithm

[91] replaces the second order information in the primal update in ESOM with a BFGS approximation (*i.e.* replaces $D_i^{(k)}$ in a call to the Hessian approximation equations in Algorithm 5 with an approximation), and results in essentially a "doubly" approximate Hessian inverse. In [92] 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.

B. Convex Surrogate Methods

While the approximate Newton methods in [90], [91], [92] optimize a quadratic approximation of the augmented Lagrangian of (6), 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 [14] 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 cost function locally. A variety of surrogate functions, $U(\cdot)$, are proposed including linear, quadratic, and block-convex which allows for greater flexibility in tailoring the algorithm to individual applications. Using its surrogate of the joint cost function, each robot updates its local variables iteratively by solving its surrogate 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 [93] algorithm extends the surrogate function principles of NEXT, and proposes a variety of non-doubly stochastic weighting schemes that can be used to perform gradient averaging similar to the push-sum protocols. The authors of SONATA also show that a several configurations of the algorithm result in already proposed distributed optimization algorithms including Aug-DGM, Push-DIG, and ADD-OPT.

Applications of Distributed Sequential Convex Programming

Distributed sequential convex programming methods have been applied to pose graph optimization problems [94] using a quadratic approximation of the objective function along with Gauss-Siedel updates to enable distributed local computations among the robots. The NEXT family of algorithms have been applied to a number of learning problems where data is distributed including semi-supervised support vector machines [95], neural network training [96], and clustering [97].

Algorithm 6 NEXT

Private variables: $\mathcal{P}_i = \left(x_i^{(k)}, \tilde{x}_i^{(k)}, \tilde{\pi}_i^{(k)}\right)$
Public variables: $\mathcal{Q}_i^{(k)} = \left(z_i^{(k)}, y_i^{(k)}\right)$
Parameters: $\mathcal{R}_i^{(k)} = \left(\alpha^{(k)}, w_i, U(\cdot), \mathcal{K}\right)$
Update equations:

$$x_i^{(k+1)} = \sum_{j\in\mathcal{N}(i)} w_{ij} z_j^{(k)}$$
$$y_i^{(k+1)} = \sum_{j\in\mathcal{N}(i)} w_{ij} y_j^{(k)} + \left[\nabla f_i(x_i^{(k+1)}) - \nabla f_i(x_i^{(k)})\right]$$
$$\tilde{\pi}_i^{(k+1)} = N \cdot y_i^{(k+1)} - \nabla f_i(x_i^{(k+1)})$$
$$\tilde{x}_i^{(k+1)} = \operatorname*{argmin}_{x\in\mathcal{K}} U\left(x; x_i^{(k+1)}, \tilde{\pi}_i^{(k+1)}\right)$$
$$z_i^{(k+1)} = x_i^{(k+1)} + \alpha^{(k)}\left(\tilde{x}_i^{(k+1)} - x_i^{(k+1)}\right)$$

V. Alternating direction method of multipliers

Considering the optimization problem in (3) with only agreement constraints, we have

$$\min_{x_i \forall i\in\mathcal{V}} \sum_{i\in\mathcal{V}} f_i(x_i) \tag{7}$$
$$\text{subject to } x_i = x_j \qquad \forall (i,j)\in\mathcal{E}. \tag{8}$$

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"). In the *alternating direction method of multipliers* (ADMM), given the separability of the global cost 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 [98], [99], [100], [101]. However, the *consensus alternating direction method of multipliers* (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 [16] .

In order to achieve a distributed update of the primal and dual variables, C-ADMM alters the agreement constraints between agents with an existing communication link by introducing auxiliary primal variables to (3) (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, C-ADMM proceeds by optimizing the auxiliary primal variables, then the original primal variables, and then the dual variables as in the original formulation of ADMM. We can perform minimization with respect to the primal variables and gradient

Algorithm 7 C-ADMM

Private variables: $\mathcal{P}_i^{(k)} = y_i^{(k)}$
Public variables: $\mathcal{Q}_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = \rho$
Update equations:

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

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

Algorithm 8 SOVA

Private variables: $\mathcal{P}_i^{(k)} = y_i^{(k)}$
Public variables: $\mathcal{Q}_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = (\rho, \Phi)$
Update equations:

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

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

ascent with respect to the dual on an augmented Lagrangian that is fully distributed among the robots:

$$\mathcal{L}_a = \sum_{i \in \mathcal{V}} f_i(x_i) + y_i^\top x_i + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \|x_i - z_{ij}\|_2^2, \qquad (9)$$

where y_i represents the dual variable that enforces agreement between robot i and its communication neighbors. 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. Furthermore, we can simplify the algorithm by noting that the auxiliary primal variable update can be performed implicitly ($z_{ij}^* = \frac{1}{2}(x_i + x_j)$).

Algorithm 7 summarizes the update procedures for the local primal and dual variables of each agent. 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 DLM [102] and [103], [104] or a quadratic approximation using the Hessian such as DQM [105].

C-ADMM as presented in Algorithm 7 requires each robot to optimize over a local copy of the global decision variable x. However, many robotic problems have an 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 only requires 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. The SOVA method [106] leverages the separability of the optimization variable to achieve orders of magnitude improvement in convergence rates, computation, and communication complexity over C-ADMM methods.

In SOVA, each agent only optimizes 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. 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 8. While SOVA allows for improved

Applications of C-ADMM

ADMM has been applied to bundle adjustment and pose graph optimization problems which involve the recovery of the $3D$ positions and orientations of a map and camera [107], [108], [109], informative path planning [110]. However, these works require a central node for the dual variable updates. Other works employ the consensus ADMM variant without a central node [111] with other notable applications in target tracking [3], signal estimation [16], task assignment [112], motion planning [5], online learning [113], and parameter estimation in global navigation satellite systems [114]. Further applications of C-ADMM arise in trajectory tracking problems involving teams of robots using non-linear model predictive control [115] and in cooperative localization [116]. Applications of SOVA include collaborative manipulation [117]. C-ADMM is adapted for online learning problems with streaming data in [113].

VI. Practical Notes

In Section VII, we compare the performances of several distributed optimization methods on three benchmark problems that approximate the computational demands of typical robotics applications. However, there are several significant considerations for translating an algorithm to a fast and efficient distributed solver. Among the most important practical considerations (aside from choosing the most suitable algorithm) are parameter tuning, initialization, and communication graph topology.

A. Communication Topology

As a general rule, a more highly-connected graph facilitates faster convergence per communication iteration, regardless of

the algorithm. Just as the Fiedler value of a graph determines the rate of convergence in a pure consensus problem, the connectivity also has a direct effect on convergence in other distributed optimization problems. Fully-connected communication enables fast optimization in a multi-robot network, and even makes centralized algorithms viable alternatives to distributed optimization, depending on computational constraints and the amount of local information possessed by each robot. However, we are primarily concerned with the local, range-limited communication models that arise in practical large-scale robotics problems.

B. Parameter Tuning

The performance of each distributed algorithm that we consider is sensitive to the choice of parameters. For instance, in DGD (Algorithm 2), choosing α too large leads to divergence of the individual variables, while too small a value of α causes slow convergence. Similarly, C-ADMM (Algorithm 7) has a convergence rate that is highly sensitive to the choice of ρ, though convergence is guaranteed for all $\rho > 0$. We study the sensitivity of the convergence rate to parameter choice in each simulation in Section VII. However, the optimal parameter choice for a particular example is not prescriptive for the tuning of other implementations. Furthermore, while analytical results for optimal parameter selection are available for many of these algorithms, a practical parameter-tuning procedure is useful if an implementation does not exactly adhere to the assumptions in the literature. Appendix X-B describes the Golden Section Search (GSS) algorithm, which provides a general (centralized) procedure for parameter tuning prior to deployment of a robotic system.

C. Consensus Weights

Several distributed optimization methods, including DGD, EXTRA, DDA, NN-K, and NEXT depend not only on stepsize parameters but also on consensus weights (w in Algorithm 2) by which each robot incorporates its neighbors' variables into its update equations. In most cases, weights are assumed to be doubly stochastic (a robot's weights summed over its neighborhood is equal to one, as is the neighborhood's weights for the robot). The specific choice of weights may vary depending on the assumptions made on what global knowledge is available to the robots on the network, as discussed thoroughly in [118]. For example, [118] shows that finding pairwise symmetric weights that achieve the fastest possible consensus can be posed as a semidefinite program, which a computer with global knowledge of the network can solve efficiently. However, we cannot always assume that global knowledge of the network is available, especially in the case of a time-varying topology. In most cases, Metropolis weights facilitate fast mixing without require global knowledge. Each robot can generate its own weight vector after a single communication round with its neighbors. In fact, Metropolis weights perform only slightly sub-optimally compared to centralized optimization-based methods [119]:

$$w_{ij} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_i|,|\mathcal{N}_j|\}} & j \in \mathcal{N}_i \\ 1 - \sum_{j' \in \mathcal{N}_i} w_{ij'} & i = j \\ 0 & \text{else} \end{cases} \tag{10}$$

Simulation results for algorithms with doubly stochastic mixing matrices use Metropolis weights.

VII. Performance Comparisons in Case Studies

Many robotics problems have a distributed structure, although this structure might not be immediately apparent. In many cases, applying distributed optimization methods requires reformulating the original problem into a separable form that allows for distributed computation of the problem variables locally by each robot. This reformulation often involves the introduction of additional problem variables local to each robot with an associated set of constraints relating the local variables between the robots. We provide three examples of distributed optimization to notable robotics problems in multi-drone vehicle target tracking, drone-robot coordinated package delivery, and multi-robot cooperative mapping.

Our principal motivation in evaluating the distributed optimization methods described in Sections III-V is to assess the suitability of each method in distributed robotics applications. We measure the performances of a representative sample of the distributed optimization methods for three problem types: unconstrained least-squares, constrained convex optimization, and non-convex optimization. These classes of optimization problems are representative of a broad swath of robotic problems in estimation, localization, planning, and control.

For a given distributed optimization method, important considerations for its include the generality of the algorithm (*i.e.* the sensitivity of its performance across different scenarios to its parameters), the computational requirements (*i.e.* each robot's computation time), and the communication requirements (*i.e.* how many messages the robots must pass). We use two important metrics in evaluating performance. The first is the normalized Mean Square Error (MSE) which is computed with respect to the optimal joint solution. When evaluating distributed optimization algorithms comparisons on the basis of MSE reduction per iteration can often skew results, and does not accurately reflect the total execution time of these algorithms. For example, methods like DIGing and NEXT require a multiple communication rounds, the execution time of which is not not accounted for in a per-iteration comparison. Similarly, methods like C-ADMM require local optimization steps at each iteration, which can significantly increase computation time depending on the cost function. In robotics applications, both communication time and local computation time contribute to the overall execution time of these algorithms. Several works [120], [121] propose application-specific cost metrics that assess both communication time and computation time. In contrast to these works, we capture the trade-off between communication and computation with a single-parameter metric called Resource-Weighted Cost (RWC), which represents the total time required for a distributed optimization algorithm

to converge below an acceptable MSE given some arbitrary communication and computation capability:

$$\text{RWC}_\lambda = \frac{t_{cp} + \lambda t_{cm}}{1+\lambda}. \tag{11}$$

Here, t_{cp} is the sum of the time spent by all robots performing local computations computed by summing processor time required for local update steps across the network. The t_{cm} is the time spent by all robots communication with their neighbors which is approximated by counting the number of floating point values passed during the optimization. Finally, λ is a scaling factor that weights communication to computation resource capability. As an example, a network of robots connected via a 5G cellular network would require significantly less time to communicate the same amount of data versus the same network of robots connected via a low power radio network. Sweeping across λ corresponds to a pass over the range of possible network configurations and computational capabilities.

A. Distributed Multi-drone Vehicle Tracking

Linear least squares optimization, a convex unconstrained optimization problem, arises in a variety of robotics problems such as parameter estimation, localization, and trajectory planning. When extended to a multi-robot scenario, each of these problems can be reformulated as a distributed linear least squares optimization. In this simulation, we consider a distributed multi-drone vehicle target tracking problem in which robots connected by a communication graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each record range-limited linear measurements of a moving target, and seek to collectively estimate the target's entire trajectory. We assume that each drone can communicate locally with nearby drones over the communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The drones all share a model of the target's dynamics as

$$x_{t+1} = A_t x_t + w_t, \tag{12}$$

where $x_t \in \mathbb{R}^4$ represents the position and velocity of the target in some global frame at time t, A_t is a linear model of the targets dynamics, and $w_t \sim \mathcal{N}(0, Q_t)$ represents process noise (including the unknown control inputs to the target). At every timestep when the target is sufficiently close to a drone i (which we denote by $t \in \mathcal{T}_i$), that robot collects an observation according to the measurement model

$$y_{i,t} = C_{i,t} x_t + v_{i,t}\,, \tag{13}$$

where $y_{i,t} \in \mathbb{R}^2$ is a positional measurement, $C_{i,t}$ is the measurement model of drone i, and $v_{i,t} \sim \mathcal{N}(0, R_{i,t})$ is measurement noise. All of the drones have the same model for the prior distribution of the initial state of the target $\mathcal{N}(\bar{x}_0, \bar{P}_0)$. The global cost function is of the form

$$\begin{aligned} f(x) =& \|x_0 - \bar{x}_0\|^2_{\bar{P}_0^{-1}} + \sum_{t=1}^{T-1} \|x_{t+1} - A_t x_t\|^2_{Q_t^{-1}} \\ &+ \sum_{i \in \mathcal{V}} \sum_{t \in \mathcal{T}_i} \|y_{i,t} - C_{i,t} x_t\|^2_{R^{-1}}, \end{aligned} \tag{14}$$

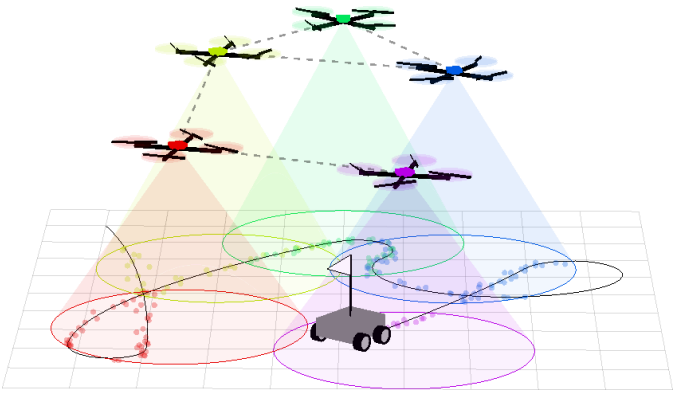


Fig. 2. A visualization of the distributed multi-drone vehicle target tracking. Each robot (colored quadrotor) records noisy observations when the target (flagged ground vehicle) is within its measurement range. The robots communicate over a network (dashed lines) to converge to the globally optimal trajectory estimate.

while the local cost function for drone i is

$$\begin{aligned} f_i(x) =& \frac{1}{N}\|x_0 - \bar{x}_0\|^2_{\bar{P}_0^{-1}} + \sum_{t=1}^{T-1} \frac{1}{N}\|x_{t+1} - A_t x_t\|^2_{Q_t^{-1}} \\ &+ \sum_{t \in \mathcal{T}_i} \|y_{i,t} - C_{i,t} x_t\|^2_{R^{-1}}. \end{aligned} \tag{15}$$

In our results, we consider only a batch solution to the problem (finding the full trajectory of the target given each robot's full set of measurements). Methods for performing the estimate in real-time through filtering and smoothing steps have been well studied, both in the centralized and distributed case [122]. An extended version of this multi-robot tracking problem is solved with distributed optimization in [3]. A rendering of a representative instance of this multi-robot tracking problem is shown in Figure 2.

In Figures 3 and 4 several distributed optimization algorithms are compared on an instance of the distributed multi-drone vehicle tracking problem. For this problem instance, 10 simulated drones seek to estimate the target's trajectory over 16 time steps resulting in a decision variable dimension of $n = 64$. We compare four distributed optimization methods which we consider to be representative of the taxonomic classes outlined in the sections above: C-ADMM, EXTRA, DIGing, and NEXT-Q. Figure 3 shows that C-ADMM and EXTRA have similar fast convergence rates per iteration while DIGing and NEXT-Q are 4 and 15 times slower respectively to converge below an MSE of 10^{-6}. The step-size hyperparameters for each method are computed by GSS (for NEXT-Q which uses a two parameter decreasing step-size we fix one according to the values recommended in [14]).

As mentioned in Section VI, tuning is essential for achieving robust and efficient convergence with most distributed optimization algorithms. Figure 4 shows the sensitivity of these methods to variation in step-size, and highlights that three of the methods (all except ADMM) become divergent for certain subsets of the tested hyper parameter space.

While MSE reduction per iteration and hyper parameter sensitivity are useful for understanding the performance of these methods on this specific problem instance, they do not provide

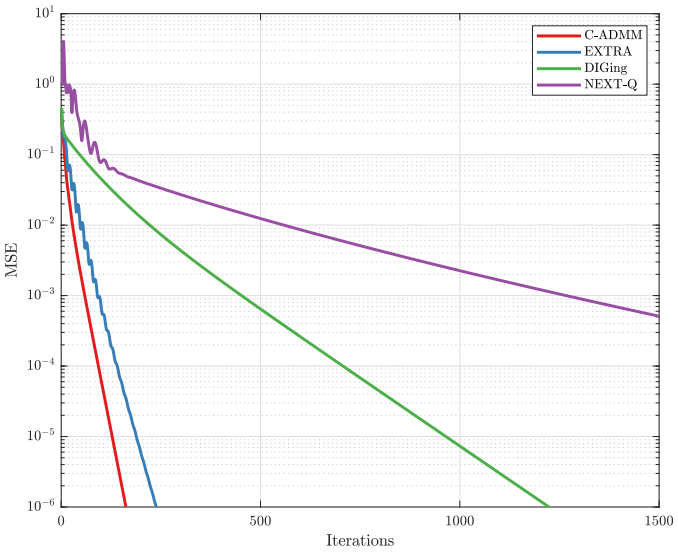


Fig. 3. MSE per iteration on a distributed multi-drone vehicle target tracking problem with $N = 10$ and $n = 64$.

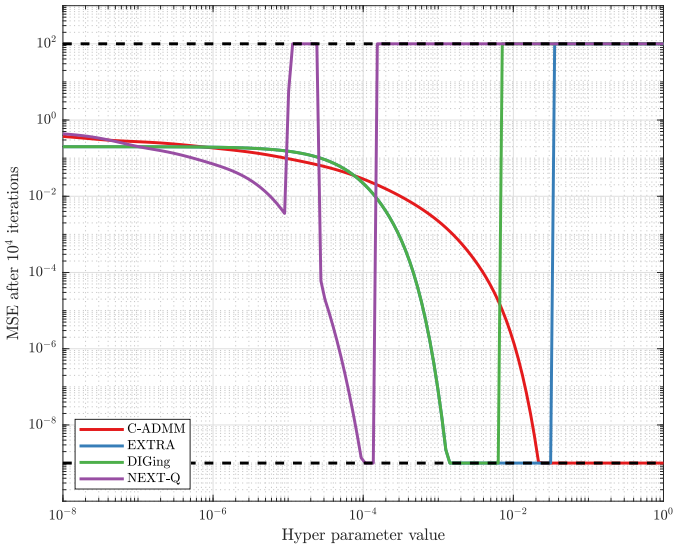


Fig. 4. step-size hyper parameter sensitivity sweep on a distributed multi-drone vehicle target tracking problem for $N = 10$ and $n = 64$. The dashed lines are thresholds for divergence (top) and convergence (bottom) in terms of MSE after 10^4 decision variable updates.

information about the scalability of these methods. Specifically, how do these algorithms scale on real networks where both communication and computation overhead are important? To answer this question we look to the RWC (11) which provides a more comprehensive analysis of the problem. In Figure 5 we show how the two fastest converging algorithms from the single instance analysis, C-ADMM and EXTRA, perform with a sweep across both problem size and RWC parameter λ.

The RWC surfaces in Figure 5 show that C-ADMM has a lower RWC compared to EXTRA at all points in the sweep, except where both λ and n are small where the algorithms have roughly the same RWC. This corresponds to scenarios with relatively small decision variables, and a network configuration where computation is costly and communication is cheap.

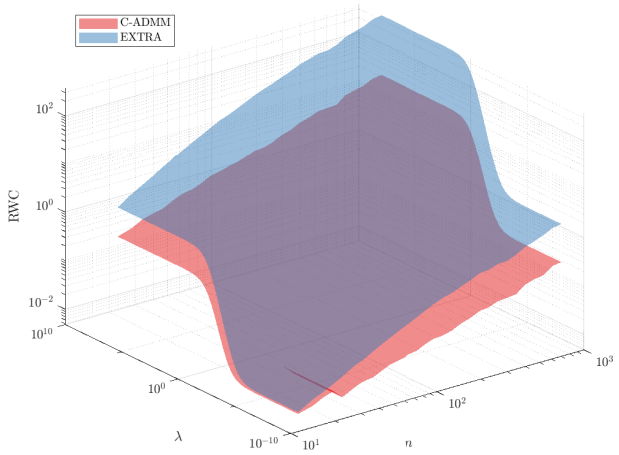


Fig. 5. Resource-weighted cost comparison of C-ADMM and EXTRA on the distributed multi-drone vehicle tracking problem. GSS was used to find the best step-sizes for each of the algorithms for each problem instance. Computation time was measured in MATLAB on a single CPU core.

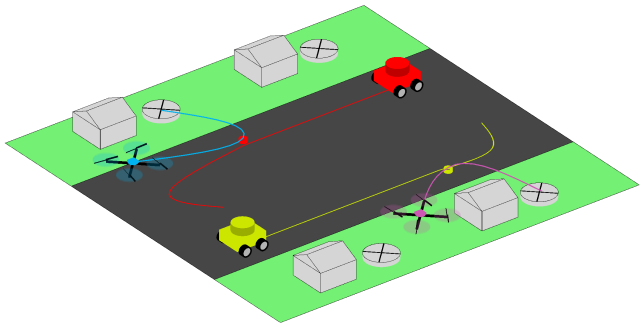


Fig. 6. Coordinated package delivery by aerial and ground robots. The aerial robots pick up packages from the ground robots for delivery to areas inaccessible to the ground robots which remain within the inner rectangular region.

B. Multi Drone-Robot Coordinated Package Delivery

Many robotics scenarios, especially in planning and control, involve constrained optimization problems, in which the decision variable must satisfy constraints such as control limits, state bounds, or load limits. In general, these problems are of the form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{N} f_i(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0. \end{aligned} \tag{16}$$

We consider the case for which the constraints $g(x) \leq 0$ and $h(x) = 0$ are convex, and each local cost function $f_i(x)$ is convex. Note that if $g(x)$ is a convex function, then $g(x) \leq 0$ is a convex constraint, and $h(x) = 0$ is a convex constraint if and only if $h(x)$ is an affine function.

The distributed subgradient methods ([11], [10], [12], [37] [33]) do not handle constrained optimization problems, and extending these methods is beyond the scope of this paper. Consequently, we compare C-ADMM to dual averaging with

push-sum (PS-DA) [65] and the distributed convex approximation method with consensus on the network gradients (NEXT-Q) in which we take a quadratic approximation of the objective function using the problem Hessian.

Consider N aerial robots delivering packages within a remote area. Each robot meets with a limited number of ground robots M at a given time to pick up packages for delivery to specified locations. In addition, the ground robots are constrained to move within certain zones. Depending on the size of the ground robots, these constraints are included to keep the robots on the roads (for larger robots) or on sidewalks (for smaller robots). We want to compute a trajectory for each robot that minimizes its energy consumption and satisfies the package pick-up, control, and state constraints. The resulting optimization problem is

$$\begin{aligned} \underset{x_a, u_a, x_g, u_g}{\text{minimize}} \quad & \sum_{i=1}^{N} u_{a,i}^T Q_{a,i} u_{a,i} + \sum_{j=1}^{M} u_{g,j}^T Q_{g,j} u_{g,j} \\ \text{subject to} \quad & x_{a,i} = x_{g,j} \quad \forall t \in \mathcal{T}_{i,j} \ \forall i, \ \forall j \\ & f_a(x_{a,i}, u_{a,i}) = 0 \quad \forall i \\ & f_g(x_{g,j}, u_{g,j}) = 0 \quad \forall j \\ & h(x_{g,j}, u_{g,j}) \leq 0 \quad \forall j \end{aligned} \tag{17}$$

where x_a and u_a denote the state and control inputs of the aerial robots while x_g and u_g denote the state and control inputs of the ground robots over the duration of the package delivery problem. Q_a and Q_g are positive definite weight matrices on the control inputs of the aerial and ground robots. The constraint in (17) ensures the aerial robots meet with the ground robots at the specified times in $\mathcal{T}_{i,j}$. The robots' states follow the dynamics represented by $f(\cdot)$. Closed-form solutions for the constrained convex optimization problem (17) do not exist.

With $x_a \in \mathbb{R}^6$ and $x_g \in \mathbb{R}^4$, we impose convex state constraints on the position and velocities of the ground robots, representing constraints on the zones which the robots can occupy. In addition, we constrain the control inputs of the ground robots and assume affine dynamics for the ground and aerial robots. The robots' delivery assignments begin and end at specified stations which we include as constraints in the optimization problem.

Representing the trajectory of all the robots as Z including their states and control inputs, we define the mean square error (MSE) of each robot's solution to the joint solution as

$$\text{MSE}(Z_1, \cdots, Z_N) = \frac{1}{N} \sum_{i=1}^{N} \|Z_i - Z^\star\|_2^2 \tag{18}$$

where $Z^\star$ represents the joint solution of the problem.

We examine the performance of C-ADMM, PS-DA, and NEXT-Q on the coordinated package delivery problem using the MSE. We show the joint solution in Figure 6 where the aerial robots pick up packages from the ground robots at locations indicated by the colored packages for delivery next to the homes inaccessible to the ground robots. The aerial robots conclude each delivery assignment at the marked locations indicated by the cross-hairs next to the homes. Similar to gradient descent methods, PS-DA shows notable sensitivity to the step-size and becomes unbounded for some values of the step-size. We selected the step-size that provided the fastest convergence. From Figure 7, C-ADMM converges faster to the joint solution compared to PS-DA and NEXT-Q which converges quickly once the agents obtain a good estimate of the problem gradients through consensus. PS-DA converges more slowly compared to the other methods.

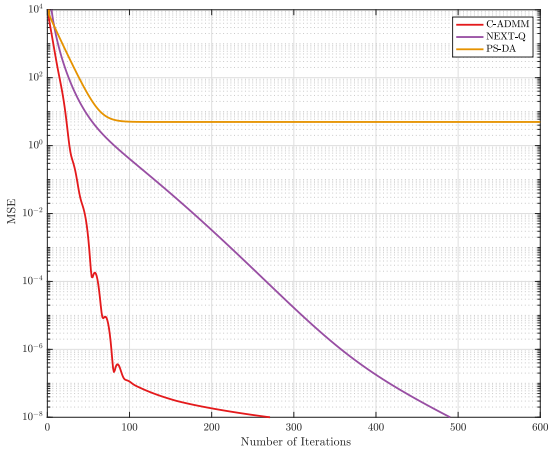


Fig. 7. Convergence error of C-ADMM, NEXT-Q, and PS-DA for the package delivery problem on range-limited graphs. PS-DA converges slowly compared to C-ADMM and NEXT-Q.

In Figure 8, we show the relative trade-off between the computation and communication overhead required by C-ADMM and NEXT-Q as the number of meeting constraints between the aerial and ground robots increases. We omit the cost of PS-DA from this figure as PS-DA converges significantly more slowly compared to the other methods. The constrained problem becomes increasingly difficult for an increasing number of meeting constraints, as reflected in Figure 8. Likewise, solving the constrained problem requires significant computation effort, and thus reducing the weight on the contribution of the computation effort to the resource-weight cost results in a lower resource-weighted cost. From Figure 8, C-ADMM achieves a lower resource-weighted cost compared to NEXT-Q.

C. Cooperative Multi-robot Mapping

In our third evaluation of distributed methods for robotics, we consider distributed nonlinear, nonconvex optimization. Many problems in robotics in robotics require optimization over nonconvex. For example, several recent papers have addressed distributed approaches to SLAM with problem-specific algorithms. In this work, we consider a milder (though still nonconvex) problem: distributed cooperative multi-robot mapping of labeled landmarks using range-only measurements. The distributed cooperative multi-robot mapping, visualized in Figure 9, consists of n robots navigating around an environment while taking measurements of the distance between their known positions and the unknown positions of m landmarks. The separable cost function with agreement constraints is given as:

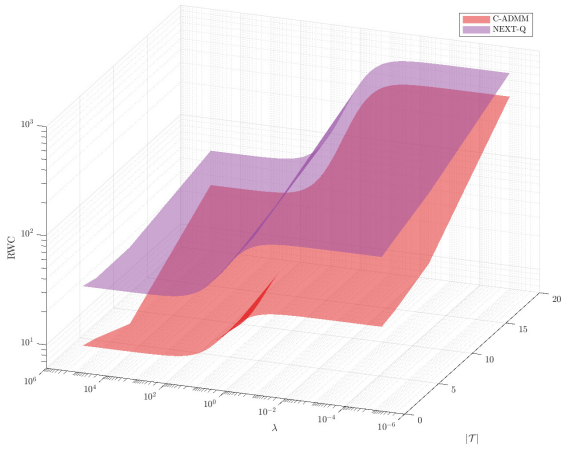


Fig. 8. Resource-weighted cost for the package delivery problem on range-limited graphs. C-ADMM attains a lower total cost considering the computation and communication required by the robots.

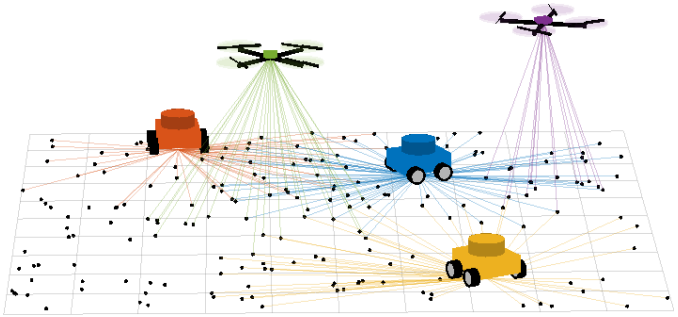


Fig. 9. A visualization of the range-only distributed cooperative multi-robot mapping problem. Heterogeneous robots record noisy range measurements to landmarks and cooperatively estimate the landmarks' positions.

$$\min_{x_{11},\ldots,x_{nm}} \sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t\in\mathcal{T}_{ik}} \frac{w_{ik}^2}{2}\left(\|p_i - x_{ik}\| - d_{ik}(t)\right)^2 \quad (19)$$
$$\text{subject to } x_{ik} = x_{jk} \quad \forall k, \forall j \in \mathcal{N}_i,$$

where x_{ik} denotes the ith robot's estimate of the location of the kth landmark.

Although the distributed methods that we consider do not have convergence guarantees for non-convex optimization problems, we have evaluated each algorithm based on the most immediate interpretation of the algorithms. In particular, we consider EXTRA and C-ADMM for this problem. The application of DGD methods to differentiable but non-convex problems such as (19) is immediate, although there is no guarantee of convergence to the global minimum.

An important consequence of solving a nonlinear least squares problem to C-ADMM is that the primal update step of the algorithm no longer consists of a single linear update. In this implementation, each robot solves the minimization step to completion (up to a specified error tolerance) before communicating with its neighbors, incurring additional computational cost per iteration. As a result, the C-ADMM update equations between communication iterations can require significantly more computation than the EXTRA update equations. Despite the higher computation cost per iteration, empirical results suggest that C-ADMM still maintains a distinct advantage over EXTRA.

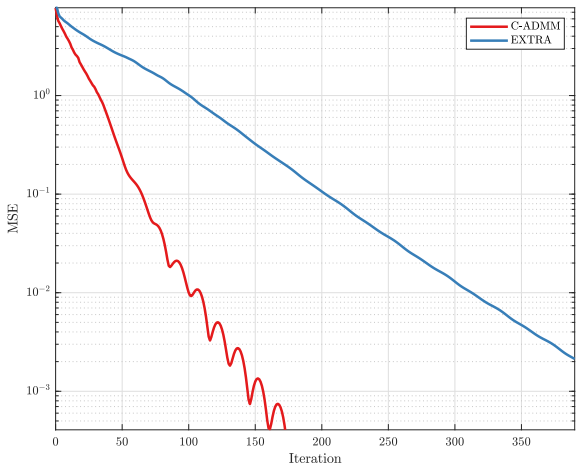


Fig. 10. Mean-square-error of robots' estimates as a function of iteration for C-ADMM and EXTRA for a fixed graph containing 20 nodes and optimal choices of step-size parameters.

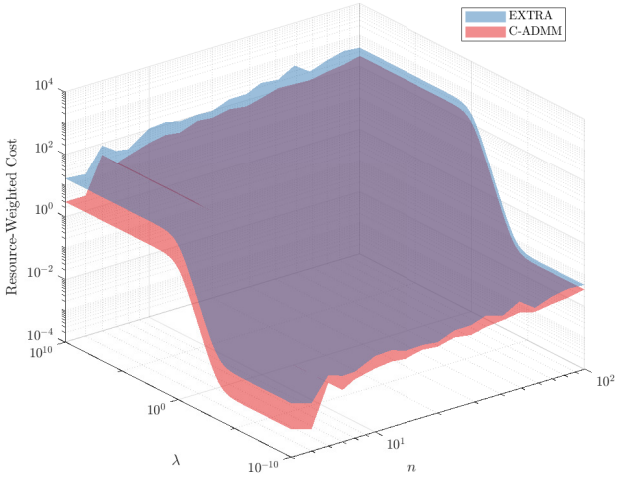


Fig. 11. Resource weighted cost of distributed optimization algorithms on a distributed cooperative multi-robot mapping problem.

VIII. Open Problems in Distributed Optimization for Robotics

Distributed optimization methods have primarily focused on solving unconstrained convex optimization problems, which constitute a notably limited subset of robotics problems. Generally, robotics problems involve non-convex objectives and constraints, which render these problems not directly amenable to many existing distributed optimization methods. For example, problems in multi-robot motion planning, SLAM, learning, distributed manipulation, and target tracking are often non-convex and/or constrained.

Both DGC methods and C-ADMM methods can be modified for non-convex and constrained problems, however few examples of practical algorithms or rigorous analyses of performance for such modified algorithms exist in the literature. Specifically, while C-ADMM is naturally amenable to constrained optimization, there are many possible ways to adapt C-ADMM to non-convex objectives, which have yet to be explored. One way to implement C-ADMM for non-convex problems is for each primal step to solve itself a non-convex 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 C-ADMM to solve each resulting quadratic problem in an inner loop. The trade-off between these two options has not yet been explored in the literature, especially in the context of non-convex problems in robotics.

Likewise, many distributed optimization methods do not consider communication between agents as an expensive resource, given that many of these methods were developed for problems with reliable communication infrastructure (e.g. multi-core computing, or computing in a hard-wired cluster). However, communication takes on greater prominence in robotics problems as robots often operate in regions with limited communication infrastructure. The absence of reliable communication infrastructure also leads to communication delays and dropped message packets. This highlights the need for research analyzing the robustness of distributed optimization methods to unreliable communication networks, and the development of new algorithms robust to such real-world communication faults.

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. In general, many distributed optimization methods involve computationally challenging procedures that require significant computational power, especially distributed methods for constrained problems. 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. As such, 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.

Finally, there are very few examples of distributed optimization algorithms implemented and running on multi-robot hardware. This leaves a critical gap in the existing literature, as the ability of these algorithms to run efficiently and robustly on robots has still not be thoroughly proven. 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.

IX. Conclusion

The field of distributed optimization provides a variety of algorithms that can address important problems for multi-robot systems. We have categorized distributed optimization methods into three broad classes—distributed gradient descent, distributed sequential convex programming, and the alternating direction method of multipliers (ADMM). We have presented a general framework for applying these methods to multi-robot scenarios, with examples in distributed multi-drone target tracking, multi-robot coordinated package delivery, and multi-robot cooperative mapping. Our empirical simulation results suggest that C-ADMM provides an especially attractive algorithm for distributed optimization in robotics problems. While distributed optimization techniques can immediately apply to several fundamental applications in robotics, important challenges remain in developing distributed algorithms for constrained, non-convex robotics problems, and algorithms tailored to the limited computation and communication resources of robot platforms.

References

[1] R. T. Rockafellar, "Monotone operators and the proximal point algorithm," *SIAM journal on control and optimization*, vol. 14, no. 5, pp. 877–898, 1976.

[2] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, Tech. Rep., 1984.

[3] O. Shorinwa, J. Yu, T. Halsted, A. Koufos, and M. Schwager, "Distributed multi-target tracking for autonomous vehicle fleets," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3495–3501.

[4] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 9649–9660.

[5] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, "A message-passing algorithm for multi-agent trajectory planning," in *Advances in neural information processing systems*, 2013, pp. 521–529.

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

[7] ——, "ZONE: Zeroth-order nonconvex multiagent optimization over networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 3995–4010, 2019.

[8] D. Hajinezhad and M. Hong, "Perturbed proximal primal–dual algorithm for nonconvex nonsmooth optimization," *Mathematical Programming*, vol. 176, no. 1-2, pp. 207–245, 2019.

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

[10] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[11] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[12] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2014.

[13] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton," *Conference Record - Asilomar Conference on Signals, Systems and Computers*, vol. 2015-April, pp. 1621–1625, 2015.

[14] P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.

[17] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[18] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[19] A. Nedić and J. Liu, "Distributed optimization for control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 77–103, 2018.

[20] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[21] 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 Processing Magazine*, vol. 37, no. 3, pp. 26–38, 2020.

[22] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, 2019.

[23] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[24] N. A. Lynch, *Distributed algorithms*. Elsevier, 1996.

[25] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, electronically available at http://coordinationbook.info.

[26] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.

[27] V. S. Mai and E. H. Abed, "Distributed optimization over directed graphs with row stochasticity and constraint regularity," *Automatica*, vol. 102, pp. 94–104, 2019.

[28] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1291–1306, 2010.

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

[30] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 33–55, 2009.

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

[32] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *2010 IEEE International Symposium on Information Theory*. IEEE, 2010, pp. 1753–1757.

[33] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[34] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

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

[36] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[37] ——, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[38] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Transactions on Signal Processing*, vol. 67, no. 17, pp. 4494–4506, 2019.

[39] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, "Exact diffusion for distributed optimization and learning—part I: Algorithm development," *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 708–723, 2018.

[40] ——, "Exact diffusion for distributed optimization and learning—part II: Convergence analysis," *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 724–739, 2018.

[41] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

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

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

[44] C. Xi, R. Xin, and U. A. Khan, "ADD-OPT: Accelerated distributed directed optimization," *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1329–1339, 2017.

[45] R. Xin and U. A. Khan, "A linear algorithm for optimization over directed graphs with geometric convergence," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 315–320, 2018.

[46] 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 Transactions on Automatic Control*, vol. 63, no. 10, pp. 3558–3565, 2018.

[47] J. Zeng and W. Yin, "ExtraPush for convex smooth decentralized optimization over directed networks," *Journal of Computational Mathematics*, vol. 35, no. 4, pp. 383–396, 2017.

[48] A. Sundararajan, B. Van Scoy, and L. Lessard, "A canonical form for first-order distributed optimization algorithms," in *American Control Conference*. IEEE, 2019, pp. 4075–4080.

[49] D. Jakovetić, "A unification and generalization of exact distributed first-order methods," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 31–46, 2018.

[50] G. Qu and N. Li, "Accelerated distributed nesterov gradient descent," *IEEE Transactions on Automatic Control*, 2019.

[51] R. Xin, D. Jakovetić, and U. A. Khan, "Distributed Nesterov gradient methods over arbitrary graphs," *IEEE Signal Processing Letters*, vol. 26, no. 8, pp. 1247–1251, 2019.

[52] Q. Lü, X. Liao, H. Li, and T. Huang, "A Nesterov-like gradient tracking algorithm for distributed optimization over directed networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[53] T. Tatarenko and B. Touri, "Non-convex distributed optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3744–3757, 2017.

[54] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.

[55] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE transactions on automatic control*, vol. 58, no. 2, pp. 391–405, 2012.

[56] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1157–1170, 2010.

[57] M. Maros and J. Jaldén, "PANDA: A dual linearly converging method for distributed optimization over time-varying undirected graphs," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6520–6525.

[58] ——, "A geometrically converging dual method for distributed optimization over time-varying graphs," *IEEE Transactions on Automatic Control*, 2020.

[59] ——, "ECO-PANDA: a computationally economic, geometrically converging dual optimization method on time-varying undirected graphs," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5257–5261.

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

[61] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," *Mathematical Programming*, vol. 180, no. 1, pp. 237–284, 2020.

[62] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical programming*, vol. 120, no. 1, pp. 221–259, 2009.

[63] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2543–2596, 2010.

[64] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling,"

IEEE Transactions on Automatic control, vol. 57, no. 3, pp. 592–606, 2011.

[65] K. I. Tsianos and M. G. Rabbat, "Distributed consensus and optimization under communication delays," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2011, pp. 974–982.

[66] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *2012 ieee 51st ieee conference on decision and control (cdc)*. IEEE, 2012, pp. 5453–5458.

[67] 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," in *MATEC Web of Conferences*, vol. 54. EDP Sciences, 2016, p. 02002.

[68] N. A. Alwan and A. S. Mahmood, "Distributed gradient descent localization in wireless sensor networks," *Arabian Journal for Science and Engineering*, vol. 40, no. 3, pp. 893–899, 2015.

[69] M. Todescato, A. Carron, R. Carli, and L. Schenato, "Distributed localization from relative noisy measurements: A robust gradient based approach," in *2015 European Control Conference (ECC)*. IEEE, 2015, pp. 1914–1919.

[70] R. Tron and R. Vidal, "Distributed image-based 3-d localization of camera sensor networks," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 901–908.

[71] ——, "Distributed computer vision algorithms," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 32–45, 2011.

[72] R. Tron, *Distributed optimization on manifolds for consensus algorithms and camera network localization*. The Johns Hopkins University, 2012.

[73] R. Tron and R. Vidal, "Distributed 3-d localization of camera sensor networks from 2-d image measurements," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3325–3340, 2014.

[74] A. Sarlette and R. Sepulchre, "Consensus optimization on manifolds," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 56–76, 2009.

[75] K.-K. Oh and H.-S. Ahn, "Formation control and network localization via orientation alignment," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 540–545, 2013.

[76] ——, "Distributed formation control based on orientation alignment and position estimation," *International Journal of Control, Automation and Systems*, vol. 16, no. 3, pp. 1112–1119, 2018.

[77] E. Montijano and A. R. Mosteo, "Efficient multi-robot formations using distributed optimization," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6167–6172.

[78] 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.

[79] T. Fan and T. Murphey, "Majorization minimization methods for distributed pose graph optimization with convergence guarantees," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5058–5065.

[80] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How, "Asynchronous and parallel distributed pose graph optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5819–5826, 2020.

[81] J. Knuth and P. Barooah, "Collaborative localization with heterogeneous inter-robot measurements by Riemannian optimization," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1534–1539.

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

[83] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed optimization via dual averaging," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 1484–1489.

[84] S. Shahrampour and A. Jadbabaie, "Exponentially fast parameter estimation in networks using distributed dual averaging," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 6196–6201.

[85] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed convex optimization on dynamic networks," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3545–3550, 2016.

[86] S. Lee, A. Nedić, and M. Raginsky, "Stochastic dual averaging for decentralized online optimization on time-varying communication graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6407–6414, 2017.

[87] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," in *2012 50th annual allerton conference on communication, control, and computing (allerton)*. IEEE, 2012, pp. 1543–1550.

[88] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[89] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network flow optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 905–920, 2013.

[90] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[91] M. Eisen, A. Mokhtari, A. Ribeiro, and A. We, "Decentralized Quasi-Newton Methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2613–2628, 2017.

[92] M. Eisen, A. Mokhtari, and A. Ribeiro, "A Primal-Dual Quasi-Newton Method for Exact Consensus Optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 23, pp. 5983–5997, 2019.

[93] Y. Sun and G. Scutari, "Distributed nonconvex optimization for sparse representation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4044–4048.

[94] 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," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1286–1311, 2017.

[95] S. Scardapane, R. Fierimonte, P. Di Lorenzo, M. Panella, and A. Uncini, "Distributed semi-supervised support vector machines," *Neural Networks*, vol. 80, pp. 43–52, 2016.

[96] S. Scardapane and P. Di Lorenzo, "A framework for parallel and distributed training of neural networks," *Neural Networks*, vol. 91, pp. 42–54, 2017.

[97] R. Altilio, P. Di Lorenzo, and M. Panella, "Distributed data clustering over networks," *Pattern Recognition*, vol. 93, pp. 603–620, 2019.

[98] H. Terelius, U. Topcu, and R. M. Murray, "Decentralized multi-agent optimization via dual decomposition," *IFAC proceedings volumes*, vol. 44, no. 1, pp. 11 245–11 251, 2011.

[99] B. Houska, J. Frasch, and M. Diehl, "An augmented Lagrangian based algorithm for distributed nonconvex optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1101–1127, 2016.

[100] N. Chatzipanagiotis, D. Dentcheva, and M. M. Zavlanos, "An augmented Lagrangian method for distributed optimization," *Mathematical Programming*, vol. 152, no. 1-2, pp. 405–434, 2015.

[101] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *52nd IEEE conference on decision and control*. IEEE, 2013, pp. 3671–3676.

[102] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[103] T.-H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2014.

[104] 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, 2019.

[105] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "DQM: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5158–5173, 2016.

[106] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 3619–3626.

[107] R. Zhang, S. Zhu, T. Fang, and L. Quan, "Distributed very large scale bundle adjustment by global camera consensus," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 29–38.

[108] A. Eriksson, J. Bastian, T.-J. Chin, and M. Isaksson, "A consensus-based framework for distributed bundle adjustment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1754–1762.

[109] 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 *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1349–1356.

[110] S.-S. Park, Y. Min, J.-S. Ha, D.-H. Cho, and H.-L. Choi, "A distributed ADMM approach to non-myopic path planning for multi-target tracking," *IEEE Access*, vol. 7, pp. 163 589–163 603, 2019.

[111] K. Natesan Ramamurthy, C.-C. Lin, A. Aravkin, S. Pankanti, and R. Viguier, "Distributed bundle adjustment," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2146–2154.
[112] R. Haksar, O. Shorinwa, P. Washington, and M. Schwager, "Consensus-based ADMM for task assignment in multi-robot teams," in *International Symposium on Robotics Research*, 2019.
[113] H. F. Xu, Q. Ling, and A. Ribeiro, "Online Learning over a Decentralized Network Through ADMM," *Journal of the Operations Research Society of China*, vol. 3, no. 4, pp. 537–562, 2015.
[114] A. Khodabandeh and P. Teunissen, "Distributed least-squares estimation applied to GNSS networks," *Measurement Science and Technology*, vol. 30, no. 4, p. 044005, 2019.
[115] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of multiple vessels via distributed nonlinear model predictive control," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 2523–2528.
[116] S. Kumar, R. Jain, and K. Rajawat, "Asynchronous optimization over heterogeneous networks via consensus ADMM," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 114–129, 2016.
[117] O. Shorinwa and M. Schwager, "Scalable collaborative manipulation with distributed trajectory planning," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'20*, vol. 1. IEEE, 2020, pp. 9108–9115.
[118] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.
[119] S. Jafarizadeh and A. Jamalipour, "Weight optimization for distributed average consensus algorithm in symmetric, CCS & KCS star networks," *arXiv preprint arXiv:1001.4278*, 2010.
[120] A. S. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei, "Balancing communication and computation in distributed optimization," *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3141–3155, 2018.
[121] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2021–2031.
[122] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 5492–5498.
[123] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling *et al.*, "Numerical recipes," 1989.

X. Appendix

A. Fixed Step-size DGD General Form

As demonstrated in [48], a range of fixed-step-size DGD methods, including EXTRA [10], NIDS [38], Exact Diffusion [39], and DIGing [36] can be alternatively represented in a canonical form. In this canonical form, weights define the inclusion of several terms common to fixed-step-size DGD including the local gradient and the local decision variable history. Algorithm 9 provides a unifying template for fixed step-size distributed gradient descent (DGD) methods with the selection of the parameters α, ζ_0, ζ_1, ζ_2, and ζ_3 defining a specific DGD method. For an illustrative example, we obtain EXTRA (refer to Algorithm 3) from the parameters $\zeta_0 = \frac{1}{2}$, $\zeta_1 = 1$, $\zeta_2 = 0$, and $\zeta_3 = 0$.

Algorithm 9 Fixed Step-Size Distributed Gradient Descent

Private variables: $\mathcal{P}_i = \emptyset$
Public variables: $\mathcal{Q}_i^{(k)} = \left(x_i^{(k)}, z_i^{(k)}\right)$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3)$
Update equations:

$$\begin{aligned} x_i^{(k+1)} = x_i^{(k)} &+ \zeta_0 z_i^{(k)} - \zeta_1 \left(x_i^{(k)} - \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} \right) \cdots \\ &+ \zeta_2 \left(z_i^{(k)} - \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} z_j^{(k)} \right) \cdots \\ &- \alpha \nabla f_i \left((1-\zeta_3) x_i^{(k)} + \zeta_3 \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} \right) \\ z_i^{(k+1)} = z_i^{(k)} &- x_i^{(k)} + \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} \end{aligned}$$

Algorithm 10 Golden Section Search

function GSS(a_0, a_3, $f(\cdot)$)
  $h \leftarrow a_3 - a_0$
  $(a_1, a2) \leftarrow (a_0 + \varphi^2 h, a_0 + \varphi h)$
  while stopping criterion is not satisfied **do**
    if $f(a_1) < f(a_2)$ **then**
      $(a_3, a_2) \leftarrow (a_2, a_1)$
      $h \leftarrow \varphi h$
      $a_1 \leftarrow a_0 + \varphi^2 h$
    else
      $(a_0, a_1) \leftarrow (a_1, a_2)$
      $h \leftarrow \varphi h$
      $a_2 \leftarrow a_0 + \varphi h$
    end if
  end while
end function

B. Golden Section Search

We assume that rather than tuning parameters online using a distributed algorithm, the roboticist selects suitable parameters for an implementation before deploying a system, either using analytical results or simulation. In this case, the most general (centralized) procedure for parameter tuning involves comparing the convergence performance of the system on a known problem for different parameter values. However, while a uniform sweep of the parameter space may be effective for small problems or parameter-insensitive methods, it is not computationally efficient. Given the convergence rate of a distributed method at particular choices of parameter, *bracketing* methods provide parameter selections to more efficiently find the convergence-rate-minimizing parameter. For instance, Golden Section Search (GSS) provides a versatile approach for tuning a scalar parameter [123]. Assuming that the dependence of convergence rate on the given parameter is strictly unimodal (as demonstrated in Section VII), GSS maintains four parameter candidates and refines its list of candidates based on performance of the interior candidates. Algorithm 10 outlines the basic procedure for parameter search. We present the performance results in Section VII according to the best parameter choice using GSS on $\log_{10}(a)$ for each parameter a (typically requiring a total of 10-15 problem evaluations).