Localized and Incremental Probabilistic Inference for Large-Scale Networked Dynamical Systems

Kai Matsuka and Soon-Jo Chung, *Senior Member, IEEE*

***Abstract*—In this article, we present new algorithms for distributed factor graph optimization (DFGO) problems that arise in the probabilistic inference of large-scale networked robotic systems for both batch and real-time problems. First, for the batch DFGO problem, we derive a type of the alternating direction method of multipliers (ADMM) algorithm called the local consensus ADMM (LC-ADMM). The LC-ADMM is fully localized; therefore, the computational effort, communication bandwidth, and memory for each agent scale like $o(1)$ with respect to the *network size*. We establish two new theoretical results for the LC-ADMM: 1) exponential convergence when the objective is strongly convex and has a Lipschitz continuous subdifferential and 2) $o(1/k)$ convergence when the objective is convex and has a unique solution. We also show that the LC-ADMM allows the use of nonquadratic loss functions, such as ℓ_1-norm and Huber loss. Second, we also develop the incremental DFGO (iDFGO) algorithm for real-time problems by combining the ideas from the LC-ADMM and the Bayes tree. To derive a time-scalable algorithm, we exploit the temporal sparsity of the real-time factor graph and the convergence of the augmented factors of the LC-ADMM. The iDFGO algorithm incrementally recomputes estimates when new factors are added to the graph and is scalable with respect to both network size and time. We validate the LC-ADMM and iDFGO in simulations with examples from multiagent simultaneous localization and mapping and power grids.**

***Index Terms*—Distributed estimation, distributed optimization, multiagent systems.**

I. Introduction

LARGE-SCALE distributed estimation problems are common in a wide variety of engineering applications, such as localization of wireless sensor networks [1], [2], tracking of electrical power grids, swarm robotics [3], [4], [5], and multiagent simultaneous localization and mapping (SLAM) [6], [7]. Over the past few decades, multiple distributed algorithms have been developed, and the results are well understood for small-scale problems, in which the dimension of the problem is small. However, there still exist challenges when solving *large-scale* problems [5], [8], [9], e.g., the number of variables may grow proportionally with the size of the network when a group of sensors is tasked with variables that are spatially distributed. Since the problem dimension is large, any algorithm that requires each agent to have an identical copy of the overall state vector will suffer from the "Curse of Dimensionality" in memory, communication, and computation.

One way to address such large-scale estimation problems is to implement a *localized* algorithm where each agent is only required to keep a small local subset of variables and uses local communications to exchange information. However, most small-scale distributed algorithms that are formulated as a recursive estimator do not naturally extend to a localized algorithm. One of the primary reasons is that the prior distribution is densely correlated in the recursive formulation in general. Due to this dense correlation, the network needs to collaboratively compute the cross terms between the variables that are far from each other. Therefore, as the network size increases, the necessary computation and communication of recursive estimators become intractable.

More recently, some papers in the robotics literature proposed solving large-scale estimation problems in a localized fashion using factor graphs [7], [10], [11]. Unlike recursive estimation algorithms in which only the current state is estimated, the factor graph optimization (FGO) approach computes the maximum a posteriori (MAP) estimate of the state trajectory in a batch. Factor graphs are a family of graphical models that represent the joint probability distribution, and they effectively model the spatial sparsity structure necessary for localized algorithms. We refer to the FGO problem involving a network of distributed agents as the distributed factor graph optimization (DFGO) problem. DFGO has the potential to be broadly applicable to various types of distributed large-scale estimation problems. However, the existing algorithms have some limitations. For example, the approach may inherently assume Gaussian noise [10], be specialized to pose graphs [7], [12], or be restricted to one agent update per iteration or require prior network topology for parallel implementation [7]. Moreover, in the process of pursuing a localized and network-scalable algorithm, the batch optimization approach reintroduced the issue of scalability with respect to time that recursive algorithms did not have. Finally, while incremental FGO algorithms exist for the single-agent case [13], [14], similar work on the distributed setting is limited in the literature, and integrating the incremental solver to distributed settings is more involved. In this article, we develop a new approach to the probabilistic inference problem for large-scale

Manuscript received 23 October 2022; revised 27 March 2023; accepted 13 June 2023. Date of publication 6 September 2023; date of current version 4 October 2023. This paper was recommended for publication by Associate Editor Eduardo Montijano and Editor Paolo Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported by the Jet Propulsion Laboratory, California Institute of Technology. *(Corresponding author: Kai Matsuka.)*

The authors are with the California Institute of Technology, Pasadena, CA 91125 USA (e-mail: kmatsuka@caltech.edu; sjchung@caltech.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3297010.

Digital Object Identifier 10.1109/TRO.2023.3297010

networked systems. First, we apply a type of alternating direction method of multipliers (ADMM) algorithms [15], [16], [17] called the local consensus ADMM (LC-ADMM) to DFGO. The LC-ADMM extends the separable optimization variable ADMM (SOVA) [18], [19], which is a distributed and localized algorithm for large-scale optimization. In this article, we provide new theoretical results for explicit convergence rates under certain convex problems as well as provide useful interpretations of the algorithm in terms of factor graphs.

The LC-ADMM has multiple properties that are useful to DFGO. First, the LC-ADMM naturally allows the use of various types of convex loss functions such as ℓ_1-norm and Huber loss for sparse outlier rejection. Second, the LC-ADMM does not require computing any global topology information (e.g., graph coloring), making it suitable for ad hoc or large networks. Third, the LC-ADMM does not need to compute a summarized graph or exchange probability distribution parameters such as covariances [6]. Finally, *all* the agents run their algorithms in parallel simultaneously rather than taking turns [7], improving computational speed. Some of these properties can be seen in previous approaches to DFGO in the literature [7], [10], [12], but not simultaneously in a single framework. By applying the ideas from the ADMM literature [18], [19], we provide a new approach that has these properties.

In addition to the LC-ADMM, we also develop the incremental DFGO (iDFGO) algorithm to solve real-time DFGO problems. The iDFGO algorithm builds upon the LC-ADMM and leverages the tools we have for the single-agent incremental FGO (i.e., iSAM2 [14]). Simply applying iSAM2 to the local FGO update step of DFGO algorithms does not actually lead to a time-scalable algorithm. Instead, our iDFGO algorithm makes some unique extensions to iSAM2 such that each local FGO update step is computed in a way that is scalable in time. In numerical simulations, we simulate LC-ADMM/iDFGO on hundreds of agents and on multiagent pose graph optimization (PGO) problems to validate their scalability, convergence, optimality, and other properties.

A. Related Work

The work in this article is primarily related to three bodies of literature: distributed optimal estimation, FGO motivated by SLAM, and distributed optimization. In the following sections, we review the literature on each of these topics.

1) Distributed Optimal Estimation: The early works in distributed optimal estimation address the problem of having a large number of observers in a low-dimensional system [20], [21], [22]. Some authors developed the distributed minimum variance estimate for linear time-invariant (LTI) systems [23], and others generalized the solution to nonlinear dynamical systems with non-Gaussian distribution [24]. These approaches often use a recursive formulation with a number of consensus iterations at each time step. However, in general, they require each agent in a system with n-dimensional state vector to: 1) have an estimate vector of size n; 2) communicate an $n \times n$ matrix for the covariance inverse (or sometimes more for non-Gaussian distribution [24]); and 3) run a computation that scales with $O(n^3)$ at each time step [8]. Therefore, these algorithms do not extend well to large-scale problems in which n increases with the number of agents in the network.

Some prior works in distributed optimal estimation addressed the challenges of solving the optimal estimate for large-scale problems in a distributed fashion. Some authors developed a localized algorithm that exploits the L-banded structure of locally coupled information matrices [8]. However, their approach is specialized to linear Gaussian systems and also requires the global presorting of nodes to obtain the L-banded matrix. Wang et al. [9] applied system-level synthesis [25] to the distributed estimation of LTI systems. They achieved a local decomposition by estimating the trajectories in a batch, preserving the sparsity of the network graph. However, it is unclear how well the system-level synthesis approach [25] can be extended to a more general class of problems beyond LTI systems. Also, if the dimension of the local variables is n_i, the communication bandwidth for [8] and [9] scales like n_i^2.

Similar to [8] and [9], iDFGO can be classified as a type of optimal estimation algorithm that is distributed and localized. In contrast with the previous works, the LC-ADMM and iDFGO are not limited to LTI systems, and their communication size scales like n_i. Unlike [9], iDFGO is also scalable with respect to the length of the time horizon.

2) FGO and SLAM: SLAM is an important class of estimation problems in which the state vector is large scale, even for the single-agent case, and only gets larger for problems that involve multiple robots. The curse of dimensionality has motivated researchers to develop algorithms for SLAM that are increasingly efficient and scalable.

Among other techniques, FGO is known to be a key mathematical tool that efficiently solves SLAM problems by exploiting the sparse structure inherent to the problem [14], [26]. There has been recent work on multiple agents cooperatively building and solving the FGO problem, which we refer to as DFGO. DDF-SAM2 [6] is a distributed algorithm, where each agent computes the "summarized graph" and communicates that graph to its neighbors. However, computing the summarized graph is relatively expensive and [6] uses conservative approximations to make the problem tractable. The computation also requires "antifactors" to negate the effect of factors from the previous epoch. Another algorithm, DC2-PGO [7], is a certifiably correct algorithm for solving the multiagent PGO problem, which is a subset of FGO where all the decision variables are SE(d) and the observations consist of relative poses. DC2-PGO has the advantage of having a theoretical guarantee to converge to the global minimum under mild conditions, but because the algorithm derivation relies on the special structure of SE(d), it cannot be used to solve more generic FGO problems involving other variables (e.g., biases, velocities, and landmarks) and/or observation models. Riemannian block coordinate descent (RBCD) [7] is a distributed and localized optimization algorithm, and it is one of the subroutines within DC2-PGO. For optimization problems with manifold-constrained variables, RBCD is shown to converge to a first-order optimal solution

under some assumptions. However, RBCD's baseline algorithm only updates one agent at a time, making the algorithm slow when the number of agents is large. Its extension allows a subset of nonneighboring agents to run updates in parallel, but it requires the whole agents to know the network coloring a priori, posing additional restrictions for large-scale or ad hoc networks.

Aside from scalability with respect to the number of agents, another important aspect of the FGO is the challenge of scalability in *time*. There are many works in this area on single-agent settings. Some proposed a sliding time-window approach to solving single-agent SLAM [27]. Others proposed incremental algorithms that maintain information on the whole trajectory but only recompute a small subset of the problem most of the time [13], [14], [28], [29]. In particular, iSAM2 [14] is a notable work that reformulates the factor graphs into the Bayes trees of conditional probability factors. While one can find many works on incrementally solving the FGO in single-agent scenarios, a similar approach in multiagent settings has been limited. The authors of DDF-SAM2 [6] briefly mention that their local FGO problems can be solved using iSAM2, which is true. However, simply combining an incremental solver with a distributed algorithm does not necessarily yield an overall algorithm that is scalable in time. The algorithm may still involve a large number of variables resulting in a computational complexity effectively equivalent to solving the whole trajectory in a batch. The previous work [6] does not address these challenges associated with scalability in time.

In the context of these previous works on FGO, the LC-ADMM can be viewed as a new approach to solving the DFGO in batch. Like [6] and [7], the LC-ADMM is fully localized and scalable with network size, and all the agents run the algorithm in parallel. We show that the LC-ADMM has explicit convergence rate guarantees for convex problems. For nonconvex problems such as PGO, the LC-ADMM algorithm can be still derived, and we empirically show in simulation that the LC-ADMM converges to the centralized optimal solution quickly for a multiagent PGO problem. We also develop iDFGO, which is the first distributed and incremental algorithm that is designed to be scalable both in network size and in time. The iDFGO has a local FGO update step that incrementally recomputes the factor graphs. This step can be viewed as an extension of iSAM2 with specialized modifications.

3) Distributed Optimization and ADMM: Once the multiagent MAP estimation problem is formulated as an FGO, a distributed optimization algorithm is needed to solve it among the network of agents. In a distributed optimization problem, the agents collectively compute the solution to a coupled objective function in a distributed fashion [7], [30], [31], [32], [33]. While various algorithms exist in the literature [34], [35], [36], [37], the ADMM has become a popular tool as it converges for a broad class of problems [38], [39], [40]. In particular, the decentralized-consensus-based ADMM (DC-ADMM) [39] has been shown to be applicable to multiagent systems [41]. However, the DC-ADMM assumes that all the agents have copies of the same state vector and is, therefore, not scalable.

Other works [18], [19], [42] extended DC-ADMM-like algorithms to localized settings where each agent's objective only depends on a small subset of decision variables. These works were scalable with network size but have some limitations. The algorithm in [42] required the graph coloring to be known. The SOVA algorithm [18], [19] removed the coloring requirement of [42], making its application to multiagent robotics much more suitable. However, the theoretical convergence rate guarantees were limited to asymptotic convergence [15]. Because SOVA is an ADMM of decentralized consensus type, many of the existing results in the other ADMM literature do not apply directly [39]. Therefore, the convergence rates and conditions under which these algorithms converge remained unknown. Some works applied the ADMM to solve real-time problems by Shorinwa et al. [41].

Our work on LC-ADMM extends the results from DC-ADMM [39] and SOVA [18], [19] in two ways. First, we establish the new theoretical results on the convergence rate of LC-ADMM under two different sets of assumptions. We show that the LC-ADMM converges to the optimal solution at $o(1/k)$ rate when the objective function is convex and the problem has a unique solution (see Theorem 2) and at an exponential rate when the objective is strongly convex and has a Lipschitz continuous subdifferential (see Theorem 3). In both the cases, the objective may be nondifferentiable, and the same convergence is given in the presence of local affine equality constraints. Theorem 3 is a generalization of the result of [39] to localized, nondifferentiable, and affine equality-constrained settings, and Theorems 2 and 3 provide, with additional assumptions, faster rates than the asymptotic convergence given in [18]. Second, our work also provides new perspectives on the LC-ADMM in the context of DFGO. During the LC-ADMM step in which each agent solves its local optimization in parallel, the local optimization problem has an intuitive interpretation as FGO. This perspective enables the development of efficient algorithms using tools from the FGO literature.

B. Summary of Contributions and Article Organization

The contributions of this article are as follows.

1) We propose a new approach toward solving DFGO using the LC-ADMM. In addition to the scalability with respect to the number of agents, the LC-ADMM has various advantages. The algorithm can naturally incorporate convex robust loss functions, such as ℓ_1-norm and Huber losses; agents do not need to know information about global graph topology; and there is an intuitive interpretation of the LC-ADMM algorithm steps in terms of factor graphs.
2) Using LC-ADMM as a backbone, we develop the iDFGO algorithm for real-time problems. The iDFGO algorithm can incrementally recompute a subset of the local problem rather than solving the optimization over the whole trajectory. The iDFGO algorithm is scalable both in network size and time.
3) We show two new theoretical results on the convergence rate of LC-ADMM. First, Theorem 2 shows $o(1/k)$

convergence when the objective is convex and has a unique solution. Second, Theorem 3 shows that the LC-ADMM converges exponentially when the objective function is strongly convex and has a Lipschitz continuous subgradient. These convergence rates are shown to hold even in the presence of additional local affine equality constraints. Theorem 3 can be seen as a generalization of results in [39].

4) We perform numerical validations of LC-ADMM and iDFGO using examples from power grid monitoring and multiagent PGO. To the best of the author's knowledge, a unified formulation of the distributed outlier rejection problem for a networked dynamical system with more than a hundred agents is demonstrated for the first time. We also empirically validate LC-ADMM and iDFGO for multiagent PGO problems using a benchmark dataset and compare our results against the state-of-the-art distributed PGO algorithm.

The rest of this article is organized as follows. Section II considers the DFGO problem for one particular time horizon and derives the LC-ADMM algorithm. Section III establishes theoretical guarantees of convergence as well as some of the other mathematical properties of LC-ADMM. Section IV extends the LC-ADMM to real-time problems and develops the iDFGO algorithm. Section V gives the numerical validation of the algorithms in practical examples. Finally, Section VI concludes this article.

II. Problem Statement

We describe a locally coupled dynamical system as one of the motivating examples in Section II-A. We define the locally coupled FGO that arises from the probabilistic inference of multiagent systems in Section II-B. We reformulate the locally coupled FGO as local consensus optimization in Section II-C and then introduce the LC-ADMM algorithm to solve that problem in Section II-D. Finally, we discuss some properties of LC-ADMM, such as algorithm complexity in Section II-E and variable connectivity in Section II-F.

Prior to defining the problem statement, we first introduce some of the notations that will be used throughout this article. The norm notation $\|\cdot\|$ without any subscript denotes the ℓ_2-norm. $\|\cdot\|_F$ denotes the Frobenius norm. The weighted norm for some square matrix $Q \in \mathbb{R}^{n\times n}$ is defined as $\|x\|_Q = x^\top Q x$. The maximum singular value of the matrix $Q \in \mathbb{R}^{m\times n}$ is denoted as $\sigma_{\max}(Q)$, and the minimum nonzero singular value is denoted as $\tilde{\sigma}_{\min}(Q)$. The identity matrix of dimension $N \times N$ is denoted as I_N. The set of positive integers is denoted as $\mathbb{Z}_+$. Given a set $\mathcal{S}$, its index set is denoted by $\mathcal{I}(\mathcal{S})$, such that $\mathcal{S} = \{s_i \,|\, i \in \mathcal{I}(\mathcal{S})\}$. The cardinality of a set $\mathcal{S}$ is denoted as $|\mathcal{S}|$. A Cartesian product of sets is denoted as $\mathcal{S}_1 \times \mathcal{S}_2$ or $\prod_i \mathcal{S}_i$.

A network of agents is modeled as an undirected graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ is the set of agents and $\mathcal{E} \subseteq \mathcal{A} \times \mathcal{A}$ is the set of edges. We say (i, j) is in the set of edges $\mathcal{E}$ *iff* the ith and the jth agents are connected. The set of neighbors for the ith agent is given by $\mathcal{N}^i = \{j \in \mathcal{A} \,|\, (i, j) \in \mathcal{E}\}$, and the closed set of neighbors is defined as $\bar{\mathcal{N}}^i \triangleq \mathcal{N}^i \cup \{i\}$. We refer to $\mathcal{G}$ as the *physical graph*, in order to distinguish it from the other types of graphs used in this article.

A. Motivating Example

A class of problems that motivates the development of LC-ADMM and iDFGO is estimating the state of a network of robots or locally coupled dynamical systems. Suppose that the state of agent i at time τ is denoted as $r_{i,\tau}$, and agent i makes observations $y_{i,\tau} \in \mathbb{R}^{m_i}$. The dynamical system of a group of locally coupled robots is given by

$$\begin{aligned} r_{i,\tau} &= a_{i,\tau}(r_{\bar{\mathcal{N}}^i,\tau-1}) + w_{i,\tau}, \quad \tau \in \mathbb{Z}_+, i \in \mathcal{A} \\ y_{i,\tau} &= c_{i,\tau}(r_{\bar{\mathcal{N}}^i,\tau}) + v_{i,\tau}, \quad \tau \in \mathbb{Z}_+, i \in \mathcal{A} \end{aligned} \tag{D1}$$

where $r_{\bar{\mathcal{N}}^i,\tau} \triangleq \{r_{j,\tau} \,|\, j \in \bar{\mathcal{N}}^i\}$ denotes the set of states at time τ in the closed neighborhood and the random noise $w_{i,\tau} \in \mathbb{R}^{p_i}$ and $v_{i,\tau} \in \mathbb{R}^{q_i}$ are assumed to be independent. Each agent's dynamics and measurement ($a_{i,\tau}$ and $c_{i,\tau}$, respectively) depend on its states as well as those of its neighbors. The system in (D1) models a variety of local interactions such as the downwash interaction of quadcopters [43] or relative range and bearing between robots.

The team of robots is tasked with collaboratively computing the MAP estimate given the set of all the observations available up to the current time t. In this example, the set of the state to be estimated is $\mathcal{X}_t = \{r_{i,\tau} \,|\, i \in \mathcal{A}, \tau = [0, t]\}$, and the set of measurements available is $\mathcal{Y}_t = \{y_{i,\tau} \,|\, i \in \mathcal{A}, \tau = [0, t]\}$. The MAP estimate at t is given by

$$\mathcal{X}_{t,*} \triangleq \arg\max_{\mathcal{X}_t} p(\mathcal{X}_t|\mathcal{Y}_t) = \arg\max_{\mathcal{X}_t} p(\mathcal{Y}_t|\mathcal{X}_t)p(\mathcal{X}_t) \tag{1}$$

where $p(\mathcal{X}_t)$ is the prior distribution and $p(\mathcal{Y}_t|\mathcal{X}_t)$ is the likelihood function. The prior distribution represents the probabilistic relations according to the dynamics of the system, while the likelihood function describes the observations made by the network. The rest of Sections II and III focus on solving (1) for a fixed time horizon $\mathcal{T}(t)$ using LC-ADMM. For readability, the subscript t from $\mathcal{X}_t$ and $\mathcal{Y}_t$ is dropped in these sections.

B. FGO of Multiagent Systems

The optimization in (1) can be viewed an FGO problem. Let $\mathcal{X} = \{x_s\}$ be the variables to be estimated and $\mathcal{Y} = \cup_{i\in\mathcal{A}}\mathcal{Y}^i$ be the set of measurements, where $\mathcal{Y}^i$ is ith agent's observations. The definitions of $\mathcal{X}$ and $\mathcal{Y}^i$ here can be more general than those we considered in example (D1).[1] Each element x_s is referred to as a *variable node* where the subscript s is used to index the variable nodes in $\mathcal{X}$ throughout this article.

We assume that the joint probability distribution in (1) can be factored as

$$p(\mathcal{X}, \mathcal{Y}) = \prod_{i\in\mathcal{A}} \prod_{\phi_f\in\mathcal{F}_i} \phi_f(\mathcal{X}^{\phi_f})$$

[1]For example, $\mathcal{X}$ may include nonrobot states such as landmarks or time-varying target states and $\mathcal{Y}$ may include loop closure observations.

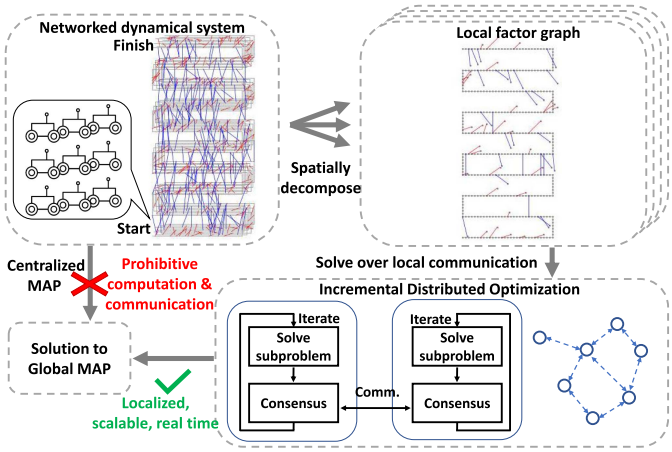

Fig. 1. Our approach to solving the locally coupled multiagent FGO problem of large-scale networks. The LC-ADMM algorithm solves the DFGO problem in a localized fashion in batch. The iDFGO algorithm solves the real-time problem in an incremental fashion.

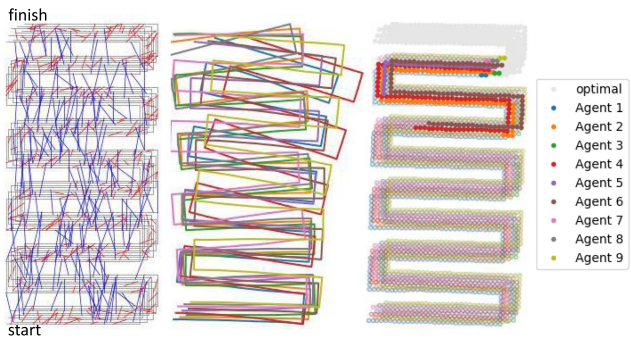

Fig. 2. Multiagent pose graph optimization solved via iDFGO. Left: Simulated trajectories of nine robots traversing the environment in a formation. Noisy data contain odometry (gray), interagent observations at a given time (red), and self/interagent loop closure constraints (blue). Center: Individual estimates without any collaboration result in large errors. Right: Nine robots collaboratively estimate the trajectories using iDFGO (colored). The filled dots represent poses that are recomputed incrementally in this time frame and nonfilled dots represent poses that were not modified. Updating only a subset of the trajectory facilitates scalability with respect to time.

where each ϕ_f is an independent probability distribution depending on a subset of variables $\mathcal{X}^{\phi_f} \subset \mathcal{X}$. The set $\mathcal{F}_i = \{\phi_f\}$ is the set of *local factors* corresponding to the ith agent. It contains the probability distributions pertaining to the ith agent, and we assume $\mathcal{F}_i \cap \mathcal{F}_j = \varnothing$ for $\forall i, j \in \mathcal{A}$, $i \neq j$. The set of all the factors in the network is defined as $\mathcal{F} \triangleq \cup_{i\in\mathcal{A}} \mathcal{F}_i$. The *factor graph* modeling the distribution $p(\mathcal{X}, \mathcal{Y})$ is defined as the bipartite graph $\mathcal{G}_F = (\mathcal{X}, \mathcal{F}, \mathcal{E}_F)$ consisting of variable nodes $\mathcal{X}$, factor nodes $\mathcal{F}$, and edges $\mathcal{E}_F$. The graph $\mathcal{G}_F$ is said to have an edge $(\phi_f, x_s) \in \mathcal{E}_F$ *iff* $\phi_f \in \mathcal{F}$ depends on $x_s \in \mathcal{X}$. The factor graph for the example in (D1) is shown on the left diagram in Fig. 3. As shown in Fig. 3, the factor graphs are often characterized by both spatial and temporal sparsity.

The local factors for the ith agent, $\mathcal{F}_i$, depend on the *local subset of variable nodes*, which is defined as

$$\mathcal{X}^{(i)} \triangleq \{x_s \in \mathcal{X} \,|\, (\phi_f, x_s) \in \mathcal{E}_F \text{ for some } \phi_f \in \mathcal{F}_i\} \subseteq \mathcal{X}.$$

Using this definition, the MAP estimation problem in (1) can be rewritten as

$$\min_{\mathcal{X}} \quad \sum_{i\in\mathcal{A}} f_i\left(\mathcal{X}^{(i)}\right) \tag{P1}$$

where each $f_i(\mathcal{X}^{(i)}) \triangleq -\sum_{f\in\mathcal{I}(\mathcal{F}_i)} \log(\phi_f(\mathcal{X}^{\phi_f}))$ is the objective. For large-scale estimation problems that involve many agents, we often have $|\mathcal{X}^{(i)}| \ll |\mathcal{X}|$, and $|\mathcal{X}^{(i)}|$ is independent of the network size. In other words, if the optimization problem in (P1) could be decoupled, the problem that each agent solves will be small. However, the objective in (P1) is still locally coupled because $\mathcal{X}^{(i)} \cap \mathcal{X}^{(j)}$ is not empty for $i \neq j$ in general.

The local coupling of the factor graph for (P1) can be visualized on the left diagram of Fig. 3. Variables are shown in circles and factors relating to variables are shown in squares. Local coupling is introduced by a subset of variables that are shared by the factors of multiple robots. If one was to solve (P1) jointly using standard FGO solvers (i.e., g2o [44] and GTSAM [14]), it will not scale with respect to the number of robots in the network.

C. Local Consensus Reformulation

Instead of solving (P1) directly, we solve a closely related *local consensus optimization problem* (P2) that allows us to partially decouple (P1), which can be later solved by LC-ADMM. As we shall see, (P1) and (P2) are equivalent under certain conditions. To develop (P2), let us first denote the ith agent's estimate for a variable node x_s as $x_s^{(i)}$ and define the set of local estimates[2] with respect to agent i as

$$\hat{\mathcal{X}}^{(i)} \triangleq \{x_s^{(i)} \,|\, s \in \mathcal{I}(\mathcal{X}^{(i)})\}. \tag{2}$$

Then, the local consensus optimization problem is written as

$$\begin{aligned} \min_{X,Z} \quad & \sum_{i\in\mathcal{A}} f_i(\hat{\mathcal{X}}^{(i)}) \\ \text{subject to} \quad & \begin{matrix} x_s^{(i)} - z_s^{(ij)} = 0 \\ x_s^{(j)} - z_s^{(ij)} = 0 \end{matrix} \quad \forall s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}. \end{aligned} \tag{P2}$$

Each $z_s^{(ij)}$ is shared between agents i and j only and its role is to enforce the equality between $x_s^{(i)}$ and $x_s^{(j)}$. The set $\mathcal{I}^{ij} \triangleq \mathcal{I}(\hat{\mathcal{X}}^{(i)}) \cap \mathcal{I}(\hat{\mathcal{X}}^{(j)})$ is the index set for all the variables shared between agents i and j. In the new problem, the decision variables are

$$X \triangleq \cup_{i\in\mathcal{A}} \hat{\mathcal{X}}^{(i)} = \{x_s^{(i)} \,|\, \forall x_s^{(i)} \in \hat{\mathcal{X}}^{(i)}, \forall i \in \mathcal{A}\}$$

$$Z \triangleq \{z_s^{(ij)} \,|\, s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}\}.$$

The visual representation of (P2) can be seen in the center diagram of Fig. 3. Compared to the factor graph representation of (P1) shown on the left, each agent now has copies of the shared variables. For each shared variable, the consistency between the multiple copies is enforced by the equality constraints as given in (P2).

[2]Later in Section II-F, we discuss how the definition of $\hat{\mathcal{X}}^{(i)}$ can be optionally modified to include additional variables. However, unless mentioned otherwise, we assume that the definition of $\hat{\mathcal{X}}^{(i)}$ is as given in (2) (i.e., $\mathcal{I}(\hat{\mathcal{X}}^{(i)}) = \mathcal{I}(\mathcal{X}^{(i)})$).

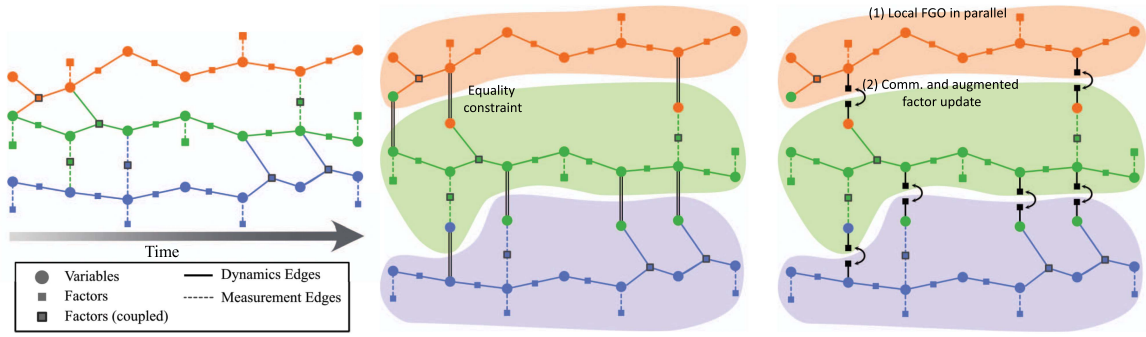

Fig. 3. Visual representation of factor graphs for estimation problem involving three robots: orange, green, and blue. Left: locally coupled multiagent FGO in (P1). Center: local consensus optimization problem (P2) replaces local coupling with equality constraints. Right: the LC-ADMM solves (P2) by iterating: 1) all the robots solving local augmented FGO in parallel and 2) neighboring robots exchange information and update the pseudo measurements of the augmented factors.

Problems (P1) and (P2) are equivalent when a condition called variable connectivity [42] is satisfied. For each $s \in \mathcal{I}(\mathcal{X})$, we define the set of *codependent agents* as

$$\mathcal{A}_s \triangleq \{i \in \mathcal{A} \,|\, x_s^{(i)} \in \hat{\mathcal{X}}^{(i)}\}.$$

In other words, agent $i \in \mathcal{A}$ is part of $\mathcal{A}_s$ *iff* it has a local estimate for x_s. The induced subgraph of the physical graph $\mathcal{G}$ with respect to the variable node x_s is the undirected graph $\mathcal{G}_s \triangleq (\mathcal{A}_s, \mathcal{E}_s)$, where $\mathcal{E}_s \subseteq \mathcal{E}$ is given by

$$\mathcal{E}_s \triangleq \{(i,j) \in \mathcal{E} \,|\, i \in \mathcal{A}_s \text{ and } j \in \mathcal{A}_s\}.$$

To establish the equivalence of (P1) and (P2), we make the following assumption on the connectivity of the network.

Assumption 1 (Variable Connectivity [42]): For all $x_s \in \mathcal{X}$, the corresponding undirected induced subgraph $\mathcal{G}_s$ is connected.

Section II-F discusses the various implications of Assumption 1. Equipped with Assumption 1, we have the following lemma.

Lemma 1 (Equivalence of (P1) and (P2)): Suppose that Assumption 1 holds. Suppose $X_* = \cup_{i\in\mathcal{A}} \hat{\mathcal{X}}_*^{(i)}$, where $\hat{\mathcal{X}}_*^{(i)} = \{x_{s,*}^{(i)} \,|\, s \in \mathcal{I}(\hat{\mathcal{X}}^{(i)})\}$, and $Z_* = \{z_{s,*}^{(ij)} \,|\, s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}\}$ are the feasible solutions of (P2). Then, there exists a feasible solution $\mathcal{X}_* = \{x_{s,*} \,|\, s \in \mathcal{I}(\mathcal{X})\}$ to (P1), such that

$$x_{s,*} = x_{s,*}^{(i)} \quad \forall s \in \mathcal{I}(\hat{\mathcal{X}}^{(i)}), i \in \mathcal{A}$$
$$x_{s,*} = z_{s,*}^{(ij)} \quad \forall s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}.$$

Proof: Consider $x_s \in \mathcal{X}$ for which multiple agents are codependent agents ($|\mathcal{A}_s| \geq 2$). The feasible solution to (P2) satisfies the equality constraints in (P2); we have $x_{s,*}^{(i)} = x_{s,*}^{(j)} = z_{s,*}^{(ij)}$ for each $(i,j) \in \mathcal{E}$, $i,j \in \mathcal{A}_s$. We also have that the induced subgraph $\mathcal{G}_s = (\mathcal{A}_s, \mathcal{E}_s)$ is connected for variable x_s by Assumption 1, so all the local estimates for x_s have identical values. This holds true for an arbitrary $x_s \in \mathcal{X}$ with $|\mathcal{A}_s| \geq 2$. By renaming $x_s^{(i)} \to x_s$ for $\forall i \in \mathcal{A}_s$ for each $x_s \in \mathcal{X}$, we have that (P2) is equivalent to (P1). ■

Lemma 1 shows that if we solve (P2), then each solution $\hat{\mathcal{X}}_*^{(i)}$ is a subset of the solution to the original centralized MAP estimation problem (P1). The next section discusses how to solve (P2) in a distributed fashion using LC-ADMM.

D. LC-ADMM Algorithm

We now introduce the LC-ADMM algorithm, which is shown to converge to the global optimal solution (P2) in convex settings. We consider a generalized version of (P2) that additionally includes local affine equality constraints on each $\hat{\mathcal{X}}^{(i)}$

$$\begin{aligned} \min_{X,Z} \quad & \sum_{i\in\mathcal{A}} f_i(\hat{\mathcal{X}}^{(i)}) \\ \text{subject to} \quad & D_i\hat{\mathcal{X}}^{(i)} = E_i \quad \forall i \in \mathcal{A} \\ & \left.\begin{aligned} x_s^{(i)} - z_s^{(ij)} &= 0 \\ x_s^{(j)} - z_s^{(ij)} &= 0 \end{aligned}\right\} \quad \forall s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}. \end{aligned} \tag{P2a}$$

This can be written more succinctly as an ADMM form [15]

$$\begin{aligned} \min_{X,Z} \quad & F(X) \\ \text{subject to} \quad & AX + BZ = C \end{aligned} \tag{P2b}$$

where $F(X) \triangleq \sum_{i\in\mathcal{A}} f_i(\hat{\mathcal{X}}^{(i)})$. Matrices A, B, and C are given by $A \triangleq [A_1; A_2; A_3]$, $B \triangleq [-I; -I; 0]$, and $C \triangleq [0; 0; C_3]$, respectively. The matrix A_1 (respectively, A_2) is defined such that the equality constraint $A_1X - Z = 0$ (respectively, $A_2X - Z = 0$) is a collection of consensus constraints where each block row corresponds to $x_s^{(i)} = z_s^{(ij)}$ (respectively, $x_s^{(j)} = z_s^{(ij)}$) for some $s \in \mathcal{I}^{ij}$ and $(i,j) \in \mathcal{E}$. $A_3X = C_3$ is defined such that the ith block row corresponds to $D_i\mathcal{X}^{(i)} = E_i$ for agent i.

The augmented Lagrangian function for (P2b) is defined as

$$\begin{aligned} & L_\beta(X, Z, W) \\ & = F(X) + \langle W, AX + BZ - C\rangle + \frac{\beta}{2}\|AX + BZ - C\|^2 \end{aligned}$$

where W is the Lagrange multiplier and $\beta > 0$ is the scalar coefficient of the augmented terms [15]. Then, the ADMM attempts to solve (P2b) using the following iterative algorithm:

$$\begin{aligned} X_{k+1} &= \arg\min_X L_\beta(X, Z_k, W_k) \\ Z_{k+1} &= \arg\min_Z L_\beta(X_{k+1}, Z, W_k) \\ W_{k+1} &= W_k + \beta(AX_{k+1} + BZ_{k+1} - C). \end{aligned} \tag{3}$$

The Lagrange multiplier W can be split into three blocks: $W = [Y; Y'; V]$. If $w_s^{(ij,i)}$ (respectively, $w_s^{(ij,j)}$) is the Lagrange multiplier corresponding to $x_s^{(i)} = z_s^{(ij)}$ (respectively, $x_s^{(j)} = z_s^{(ij)}$), then Y consists of all $w_s^{(ij,i)}$ (respectively, Y' consists of all $w_s^{(ij,j)}$) $\forall s \in \mathcal{I}^{ij}, \forall (i,j) \in \mathcal{E}$. Similarly, if $\mathcal{V}^{(i)}$ is the Lagrange multiplier corresponding to $D_i \hat{\mathcal{X}}^{(i)} = E_i$, V consists of $\mathcal{V}^{(i)}$ $\forall i \in \mathcal{A}$. The iterations (3) can be rewritten in terms of their block components

$$\hat{\mathcal{X}}_{k+1}^{(i)} = \underset{\hat{\mathcal{X}}^{(i)}}{\arg\min}\, f_i(\hat{\mathcal{X}}^{(i)}) + \frac{\beta}{2}\|D_i\hat{\mathcal{X}}^{(i)} - E_i + \frac{1}{\beta}\mathcal{V}_k^{(i)}\|^2 + \sum_{j\in\mathcal{N}^i}\sum_{s\in\mathcal{I}^{ij}} \frac{\beta}{2}\|x_s^{(i)} - z_{s,k}^{(ij)} + \frac{1}{\beta}w_{s,k}^{(ij,i)}\|^2 \tag{4}$$

$$z_{s,k+1}^{(ij)} = \frac{1}{2}\left(x_{s,k+1}^{(i)} + \frac{1}{\beta}w_{s,k}^{(ij,i)} + x_{s,k+1}^{(j)} + \frac{1}{\beta}w_{s,k}^{(ij,j)}\right) \tag{5}$$

$$w_{s,k+1}^{(ij,i)} = w_{s,k}^{(ij,i)} + \beta(x_{s,k+1}^{(i)} - z_{s,k+1}^{(ij)}) \tag{6}$$

$$\mathcal{V}_{k+1}^{(i)} = \mathcal{V}_k^{(i)} + \beta(D_i\hat{\mathcal{X}}_{k+1}^{(i)} - E_i). \tag{7}$$

Similar to other consensus ADMM algorithms, such as in [15] and [39], it is easy to show that $z_{s,k}^{(ij)} = \bar{x}_{s,k}^{(ij)} = \frac{1}{2}(x_{s,k}^{(i)} + x_{s,k}^{(j)})$ $\forall k$ if we select the initial value of $w_s^{(ij,i)}$ and $w_s^{(ij,j)}$ to be $w_{s,0}^{(ij,i)} = -w_{s,0}^{(ij,j)}$ [15]. Using this, the iterations (4)–(7) can be further simplified to the following form for implementation:

$$\hat{\mathcal{X}}_{k+1}^{(i)} = \underset{\hat{\mathcal{X}}^{(i)}}{\arg\min}\, f_i(\hat{\mathcal{X}}^{(i)}) + \frac{\beta}{2}\|D_i\hat{\mathcal{X}}^{(i)} - E_i + \frac{1}{\beta}\mathcal{V}_k^{(i)}\|^2 + \sum_{j\in\mathcal{N}^i}\sum_{s\in\mathcal{I}^{ij}} \frac{\beta}{2}\|x_s^{(i)} - \bar{x}_{s,k}^{(ij)} + \frac{1}{\beta}w_{s,k}^{(ij,i)}\|^2 \tag{8}$$

$$w_{s,k+1}^{(ij,i)} = w_{s,k}^{(ij,i)} + \beta(x_{s,k+1}^{(i)} - \bar{x}_{s,k+1}^{(ij)}) \tag{9}$$

$$\mathcal{V}_{k+1}^{(i)} = \mathcal{V}_k^{(i)} + \beta(D_i\hat{\mathcal{X}}_{k+1}^{(i)} - E_i). \tag{10}$$

Remark: These LC-ADMM iterations have a natural interpretation in terms of factor graphs. Recall that each term within $f_i(\hat{\mathcal{X}}^{(i)})$ corresponds to a factor in the local factor graph $\mathcal{G}_{Fi} = (\hat{\mathcal{X}}^{(i)}, \mathcal{F}_i, \mathcal{E}_{Fi})$. Similarly, one can view the other terms in (8) as the *augmented factors* that softly penalize the constraints. The terms, such as $E_i - \frac{1}{\beta}\mathcal{V}_k^{(i)}$ and $\bar{x}_{s,k}^{(ij)} - \frac{1}{\beta}w_{s,k}^{(ij,i)}$, represent the *pseudo measurement* of the augmented factors, and the values of the pseudo measurements are modified by (9) and (10) at each k.

With this perspective, the first part of LC-ADMM iteration (8) is referred to as the *local FGO update*. This update can be implemented as a standard FGO using single-agent solvers such as g2o or GTSAM. The second part of LC-ADMM iteration (9), (10) is referred to as the *augmented factor update*. This step only involves information exchange and summation, which can be implemented easily. We can visualize these LC-ADMM iterations in terms of the factor graph in the right diagram in Fig. 3.

Algorithm 1: Localized Consensus ADMM.

Result: $\hat{\mathcal{X}}_k^{(i)},\ i \in \mathcal{A}$

Each agent initializes $\hat{\mathcal{X}}_k^{(i)}$ and $\{w_{s,k}^{(ij,i)} \mid s \in \mathcal{I}(\hat{\mathcal{X}}^{(i)}) \cap \mathcal{I}(\hat{\mathcal{X}}^{(j)}), (i,j) \in \mathcal{E}\}$ for $k = 0$;

while $k < K$ **do**

Each $i \in \mathcal{A}$ solves its sub-problem: $\hat{\mathcal{X}}_{k+1}^{(i)} =$

$$\underset{\hat{\mathcal{X}}^{(i)}}{\arg\min}\, f_i(\hat{\mathcal{X}}^{(i)}) + \frac{\beta}{2}\|D_i\hat{\mathcal{X}}^{(i)} - E_i + \frac{1}{\beta}\mathcal{V}_k^{(i)}\|^2 + \sum_{j\in\mathcal{N}^i}\sum_{s\in\mathcal{I}^{ij}} \frac{\beta}{2}\|x_s^{(i)} - \bar{x}_{s,k}^{(ij)} + \frac{1}{\beta}w_{s,k}^{(ij,i)}\|^2;$$

For each neighboring pair $(i,j) \in \mathcal{E}$, the agents exchange $x_{s,k+1}^{(i)}$ and $x_{s,k+1}^{(j)}$ for $\forall s \in \mathcal{I}^{ij}$;

Each $i \in \mathcal{A}$ locally computes the average $\forall s \in \mathcal{I}^{ij}$, $\forall j \in \mathcal{N}^i$:

$$\bar{x}_{s,k+1}^{(ij)} = \frac{1}{2}\left(x_{s,k+1}^{(i)} + x_{s,k+1}^{(j)}\right);$$

Each $i \in \mathcal{A}$ updates its Lagrange multipliers:

$$w_{s,k+1}^{(ij,i)} = w_{s,k}^{(ij,i)} + \beta(x_{s,k+1}^{(i)} - \bar{x}_{s,k+1}^{(ij)});$$
$$\mathcal{V}_{k+1}^{(i)} = \mathcal{V}_k^{(i)} + \beta(D_i\hat{\mathcal{X}}_{k+1}^{(i)} - E_i));$$

Iterate: $k \leftarrow k+1$

end

One can execute the LC-ADMM iterations in a fully localized fashion, as summarized in Algorithm 1. First, each agent independently solves the local FGO update in parallel as (8). Next, each agent broadcasts the shared variables $x_{s,k+1}^{(i)}$ to the neighbors who need that information. Once the neighbors' estimates are received, the average $\bar{x}_{s,k+1}^{(ij)}$ is computed and the augmented factor update (9) and (10) modify the pseudo measurements. The augmented factor update also takes place locally on each agent.

E. Algorithmic Complexity of LC-ADMM

The LC-ADMM algorithm has several properties that make it suitable for a broad class of problems involving networked systems. First, the computational effort, communication bandwidth, and memory requirements of LC-ADMM have constant complexity with respect to the size of the network, so long as the $|\mathcal{X}^{(i)}| \sim O(1)$ assumption holds. Taking (D1) as an example, the communication bandwidth scales like $O(nT|\bar{\mathcal{N}}^i|)$, where $n \triangleq \dim(x_s)$ and $T \triangleq |\mathcal{T}(t)|$, and is independent of $|\mathcal{A}|$. Second, the agents do not need to share the probability distribution parameters (e.g., covariance matrix); therefore, the communication bandwidth scales only linearly with nT, not $n^2\,T^2$. This is an advantage compared to other distributed optimal estimation algorithms that require sharing the full probability distributions such as [23] and [24].

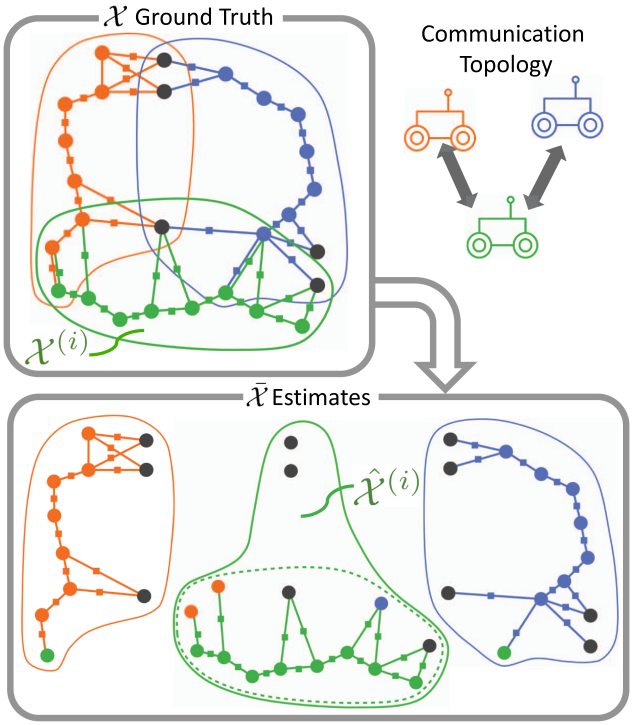

Fig. 4. Illustration of the definitions of $\mathcal{X}$, $\mathcal{X}^{(i)}$, $\hat{\mathcal{X}}^{(i)}$, and X. Colored circles represent the robot poses and the gray circles represent the landmarks. In this example, we have $\mathcal{I}(\mathcal{X}^{(i)}) = \mathcal{I}(\hat{\mathcal{X}}^{(i)})$ for $i =$ "orange," while $\mathcal{I}(\mathcal{X}^{(i)}) \subset \mathcal{I}(\hat{\mathcal{X}}^{(i)})$ for $i =$ "green." Section II-F describes what variables may be included in each $\hat{\mathcal{X}}^{(i)}$.

F. Variable Connectivity and Augmenting Set $\hat{\mathcal{X}}^{(i)}$

In some instances, the variable connectivity assumption in Assumption 1 may not be satisfied (i.e., the agents that have the local estimate for some x_s are not connected). Consider the SLAM-like example in Fig. 4. The two landmarks (shown as black circles) at the top are shared between the orange and blue robots but are not directly visible to the green robot. Since $\mathcal{X}^{(i)}$ for the green robot does not include these two landmarks and there is no direct communication link between the orange and blue robots, the variable connectivity assumption in Assumption 1 does not hold for those two landmarks.

One way to address this type of scenario is to augment the local estimate set $\hat{\mathcal{X}}^{(i)}$ with additional variables. While minimizing the number of variables in $\hat{\mathcal{X}}^{(i)}$ is desirable, the only requirement on the membership of the set $\hat{\mathcal{X}}^{(i)}$ is that it satisfies $\mathcal{I}(\mathcal{X}^{(i)}) \subseteq \mathcal{I}(\hat{\mathcal{X}}^{(i)}) \subseteq \mathcal{I}(\mathcal{X})$. In this example, the variable connectivity is satisfied by adding the two landmarks as part of $\hat{\mathcal{X}}^{(i)}$ for the green robot, as shown at the bottom of Fig. 4.

Another approach is to accept that Assumption 1 is not fully satisfied. The equality constraints for the multiple local estimates for a variable x_s are only enforced among the connected components of the induced subgraph $\mathcal{G}_s$. While (P1) and (P2) may not be strictly equivalent, the solutions for (P2) partially agree with the solution for (P1) and may still be acceptable depending on the application.

Finally, one may consider augmenting $\hat{\mathcal{X}}^{(i)}$ for reasons other than ensuring that the variable connectivity assumption holds. Consider an example where a distributed sensor network is tasked to track a common target. In this case, all the agents may include the local estimate of the target state in $\hat{\mathcal{X}}^{(i)}$, even if not all the agents make a direct observation of the target so that all agents have a copy of the target state estimate. Therefore, choosing what variables to be included $\hat{\mathcal{X}}^{(i)}$ gives users additional choices of how to constrain the problem.

III. Mathematical Properties of LC-ADMM

This section details some theoretical guarantees on the convergence rate of LC-ADMM and some properties when applied to example systems. Before discussing our results, we note that it is well known that the ADMM has (relatively slow) asymptotic convergence guarantee for a general class of convex problems without additional assumptions [15]. Since LC-ADMM is a type of ADMM, this result applies to LC-ADMM as well.

Theorem 1 (Asymptotic Convergence of ADMM [15]): Consider (P2) where the objective F is a closed, proper, convex function (not necessarily strongly convex) and the constraint $S^{(i)}$ is a convex set. Assume that the unaugmented Lagrangian has a saddle point. Then, we have the following.

1) Residual convergence: $AX_k + BZ_k \to 0$ as $k \to \infty$.
2) Objective convergence: $F(X_k) \to F(X_*)$ as $k \to \infty$.
3) Dual variable convergence: $W_k \to W_*$ as $k \to \infty$.

Proof: The proof is given in [15]. ■

In Section III-A, we establish two stronger results under additional assumptions, as follows: 1) the LC-ADMM converges at the rate of $o(1/k)$ when the problem has a unique solution (see Theorem 2) and 2) the LC-ADMM converges exponentially when the overall objective is strongly convex and has a Lipschitz continuous subgradient (see Theorem 3). Theorem 3 can be viewed as a generalization of the previous result in the literature [39] to problems that are localized, nondifferentiable, and have local equality constraints. In Section III-B, we apply Theorem 3 to a locally coupled dynamical system given in (D1) and show the relationship between observability of system and convergence. Finally, Section III-C discusses applying LC-ADMM to problems that involve inequality constraints and pose graphs optimization problems.

First, we introduce some definitions needed for establishing the convergence results.

Definition 1 (Subdifferential): The subdifferential of F at X is a closed convex set defined as

$$\partial F(X) \triangleq \bigcap_{X' \in \mathrm{dom} F} \{g \,|\, F(X') \geq F(x) + g^\top (X' - X)\}. \tag{11}$$

Definition 2 (Lipschitz Continuous Subdifferential): The subdifferential of a convex function F is *Lipschitz continuous* with parameter $L_F > 0$ *iff* for all $X, X' \in \mathrm{dom} F$ and for all $g \in \partial F(X)$ and $g' \in \partial F(X')$, we have

$$L_F \|X - X'\| \geq \|g - g'\|. \tag{12}$$

Definition 3 (Strong Convexity): A convex function F is *strongly convex* with parameter $m_F > 0$ if for all $X, X' \in \mathrm{dom} F$ and $g \in \partial F(X)$ and $g' \in \partial F(X')$, we have

$$m_F \|X - X'\|^2 \leq \langle X - X', g - g' \rangle. \tag{13}$$

A. Convergence Guarantees of LC-ADMM

To prove the convergence rates of LC-ADMM, we first rewrite the algorithm in a series of equivalent forms. Using the subgradient optimality condition, x-update (4) can be written as

$$0 \in \partial_{X_{k+1}} \left(F(X_{k+1}) + \frac{\beta}{2} \|AX_{k+1} + BZ_k - C + \frac{1}{\beta} W_k\|^2 \right)$$
$$\iff -A^\top [W_k + \beta(AX_{k+1} + BZ_k - C)] \in \partial F(X_{k+1}).$$

If we define $g_{k+1} \triangleq -A^\top[W_k + \beta(AX_{k+1} + BZ_k - C)]$, the ADMM iteration in (4)–(7) can be written as

$$g_{k+1} \in \partial F(X_{k+1}) \tag{14}$$
$$B^\top [W_k + \beta(AX_{k+1} + BZ_{k+1} - C)] = 0 \tag{15}$$
$$W_{k+1} - W_k - \beta(AX_{k+1} + BZ_{k+1} - C) = 0. \tag{16}$$

By left-multiplying (16) by $A^\top$ (respectively, with $B^\top$) and subtracting from the definition of g_{k+1} [respectively, from (15)], we have

$$g_{k+1} + A^\top [W_{k+1} + \beta B(Z_k - Z_{k+1})] = 0 \tag{17}$$
$$B^\top W_{k+1} = 0. \tag{18}$$

Since $W_k = [Y_k; Y_k'; V_k]$ and $B = [-I; -I; 0]$, (18) implies $Y_{k+1} = -Y_{k+1}'$. Recall $A \triangleq [A_1; A_2; A_3]$. If we define $\overline{M} \triangleq A_1 + A_2$ and $\underline{M} \triangleq A_1 - A_2$, then (17) can be written as

$$g_{k+1} + \underline{M}^\top Y_{k+1} + A_3^\top V_{k+1} - \beta \overline{M}^\top (Z_k - Z_{k+1}) = 0. \tag{19}$$

Equation (16) can be split into three blocks of equations by using $W_k = [Y_k; Y_k'; V_k]$ again. Adding and subtracting the first and second blocks of (16) from each other, we get

$$\frac{1}{2} \overline{M} X_{k+1} - Z_{k+1} = 0 \tag{20}$$
$$Y_{k+1} - Y_k - \frac{\beta}{2} \underline{M} X_{k+1} = 0 \tag{21}$$
$$V_{k+1} - V_k - \beta(A_3 X_{k+1} - C_3) = 0. \tag{22}$$

Assume that there exist X_* and Z_* that are the unique solution[3] to (P2b). Since (P2b) is convex, closed, and proper, we also have that $(Y_k, V_k) \to (Y_*, V_*)$ as $k \to \infty$ and (Y_*, V_*) is a stationary point [15], i.e. $\partial L_\beta(X_*, Z_*, Y_*, V_*) \ni 0$. Then, (17) and (20)–(22) evaluated at this point are

$$g_* = -\underline{M}^\top Y_* + A_3^\top V_*, \quad g_* \in \partial F(X_*) \tag{23}$$
$$\frac{1}{2} \overline{M} X_* - Z_* = 0 \tag{24}$$
$$\frac{\beta}{2} \underline{M} X_* = 0 \tag{25}$$
$$A_3 X_* - C_3 = 0. \tag{26}$$

Subtracting (23)–(26) from (19)–(22), the ADMM iteration in terms of (X_k, Z_k, Y_k, V_k) can be written as

$$g_{k+1} - g_* + \underline{M}^\top (Y_{k+1} - Y_*) + A_3^\top (V_{k+1} - V_*) - \beta \overline{M}^\top (Z_k - Z_{k+1}) = 0$$
$$g_{k+1} \in \partial F(X_{k+1}), \quad g_* \in \partial F(X_*) \tag{27}$$
$$\frac{1}{2} \overline{M} (X_{k+1} - X_*) - (Z_{k+1} - Z_*) = 0 \tag{28}$$
$$Y_{k+1} - Y_k - \frac{\beta}{2} \underline{M} (X_{k+1} - X_*) = 0 \tag{29}$$
$$V_{k+1} - V_k - \beta A_3 (X_{k+1} - X_*) = 0. \tag{30}$$

We use these equations to prove the convergence rate of LC-ADMM. For convenience, we define the subset of the variables as $U_k \triangleq [Z_k; Y_k; V_k]$. Two lemmas that establish the main theorem are the following.

Lemma 2 (Contractive Sequence U_k): Suppose that F is convex, closed, and proper and that the sequence (X_k, Z_k, Y_k, V_k) is generated by the ADMM algorithm in (3). If there exists a unique solution X_* and Z_* to (P2b), then $U_k = [Z_k; Y_k; V_k]$ satisfies

$$\|U_k - U_{k+1}\|_G^2 \leq \|U_k - U_*\|_G^2 - \|U_{k+1} - U_*\|_G^2 \tag{31}$$

where the matrix $G \succ 0$ is given by

$$G \triangleq \begin{bmatrix} \beta I & 0 & 0 \\ 0 & \frac{1}{\beta} I & 0 \\ 0 & 0 & \frac{1}{2\beta} I \end{bmatrix}. \tag{32}$$

In addition, if F is strongly convex, then we have

$$m_F \|X_{k+1} - X_*\|^2 + \|U_k - U_{k+1}\|_G^2 \leq \|U_k - U_*\|_G^2 - \|U_{k+1} - U_*\|_G^2 \tag{33}$$

where $m_F > 0$ was defined in Definition 3.

Proof: The proof outline is similar to [39] with some notable differences: we generalize the result to the localized consensus case $\mathcal{I}(\hat{\mathcal{X}}^{(i)}) \subset \mathcal{I}(\mathcal{X})$; we include local equality constraints; and we allow nondifferentiable objective functions. See the Appendix for the detailed steps. ■

Equation (31) in Lemma 2 shows that the sequence $\|U_k - U_*\|_G^2$ is monotonically nonincreasing and converging because it is lower bounded by 0. This implies $\|U_{k+1} - U_k\|_G^2 \to 0$ as $k \to \infty$. Then, $U_k \to U_*$ follows the standard analysis of contraction methods.

We show the $o(1/k)$ convergence by making a summable, nonnegative, monotonic sequence argument. The proof for the following lemma is readily available in other ADMM literature such as [38] and [45], so it is omitted for brevity.

Lemma 3: If a sequence $\{a_k\} \subseteq \mathbb{R}$ satisfies: 1) $a_k \geq 0$; 2) $\sum_{k=1}^{\infty} a_k < +\infty$; and 3) a_k is monotonically nonincreasing, then we have $a_k = o(1/k)$.

[3]This is automatically satisfied if F is strongly convex as is the case for Theorem 3.

The next theorem establishes the $o(1/k)$ convergence for the LC-ADMM.

Theorem 2 ($o(1/k)$ Convergence of LC-ADMM): Assume that F is convex, closed, and proper, and that F has a unique solution. The sequence generated by (3) converges $(X_k, U_k) \to (X_*, U_*)$ and its convergence rate is given by $\|U_k - U_{k+1}\|_G^2 = o(1/k)$.

Proof: Using the first-order optimality conditions (19)–(22) for k and $k+1$, one can follow the same manipulations as in the proof for Lemma 2 to show that

$$\begin{aligned}&\langle X_{k+1} - X_k, g_{k+1} - g_k\rangle \\ &= -2(U_{k+1} - U_k)^\top G(U_{k+1} - U_{k-1}).\end{aligned}$$

Given F is convex, it follows that

$$\|U_k - U_{k-1}\|_G^2 - \|U_{k+1} - U_k\|_G^2 \geq \|U_{k+1} - U_{k-1}\|_G^2 \geq 0.$$

Therefore $\|U_{k+1} - U_k\|_G^2$ is monotonically nonincreasing.

Next, we have that $\|U_{k+1} - U_k\|_G^2$ is summable by (31) of Lemma 2. Indeed, summing both the left- and right-hand sides of (31), we have

$$\sum_{k=0}^{\infty} \|U_k - U_{k+1}\|_G^2 \leq \|U_0 - U_*\|_G^2 < +\infty. \tag{34}$$

Because $\|U_k - U_{k+1}\|_G^2$ is nonnegative, monotonically nonincreasing, and summable, we have $\|U_k - U_{k+1}\|_G^2 = o(1/k)$ by Lemma 3. Because Z_* satisfies the equality constraint and F has a unique solution, we also have $X_k \to X_*$. ■

Theorem 2 states that $U_k \to U_*$ at the rate of $o(1/k)$ when the objective is convex and has a unique solution. Next, we show that the LC-ADMM converges exponentially if we additionally assume that F is strongly convex and has Lipschitz continuous subdifferential. Before we present the result in Theorem 3, we show Lemma 4.

Lemma 4: Suppose that F is strongly convex and its subdifferential is Lipschitz continuous. For any $\mu > 1$, define a constant c such that

$$c \triangleq \min\left\{ \frac{(\mu-1)\tilde{\sigma}_{\min}^2(\underline{M})}{\mu\sigma_{\max}^2(\overline{M})}, \frac{m_F}{\frac{\beta}{4}\sigma_{\max}^2(\overline{M}) + \frac{\mu}{\beta}L_F^2\tilde{\sigma}_{\min}^{-2}(\underline{M})} \right\} > 0. \tag{35}$$

where $L_F > 0$ and $m_F > 0$ are defined in Definitions 2 and 3, respectively. Then, we have

$$c\|U_{k+1} - U_*\|_G^2 \leq m_F\|X_{k+1} - X_*\|^2 + \|U_k - U_{k+1}\|_G^2. \tag{36}$$

Proof: The proof is similar to [39] with a few notable differences: we generalize the result to the localized consensus case $\mathcal{I}(\hat{\mathcal{X}}^{(i)}) \subset \mathcal{I}(\mathcal{X})$; we include local equality constraints; and we allow nondifferentiable objective functions. See the Appendix for the proof details. ■

The proof of Lemma 4 uses the optimality of the ADMM update equations and the additional assumptions introduced. By using Lemmas 2 and 4, it is straightforward to prove the exponential stability of U_*.

Theorem 3 (Exponential Convergence of LC-ADMM): Consider the LC-ADMM that solves (P2a). Assume that F is convex, closed, and proper. If F is strongly convex and its subdifferential is Lipschitz continuous, $U_k \to U_*$ and $X_k \to X_*$ exponentially. In addition, if Assumption 1 is satisfied, the obtained solutions $\hat{\mathcal{X}}_*^{(i)}$ for each $i \in \mathcal{A}$ are the subset of $\mathcal{X}_*$ which is the solution to (P1).

Proof: Define $c > 0$ as (35) for any $\mu > 1$. After combining (33) in Lemma 2 and (36) in Lemma 4, we can write

$$\|U_{k+1} - U_*\|_G^2 \leq c_2\|U_k - U_*\|_G^2 \tag{37}$$

where $c_2 \triangleq \frac{1}{1+c}$. Since $0 < c_2 < 1$, U_k converges exponentially to U_*. Next, we observe from (33) that

$$\|X_{k+1} - X_*\|^2 \leq \frac{1}{m_F}\|U_k - U_*\|_G^2. \tag{38}$$

Since $\|U_k - U_*\|_G^2$ exponentially converges to zero, X_{k+1} exponentially converges to X_*. ■

Theorem 3 is a generalization of the convergence rate of DC-ADMM in [39]. The result is extended to problems that may involve localized consensus, local affine equality constraints, and nondifferentiable objective functions. DC-ADMM is a special case of LC-ADMM, where $\hat{\mathcal{X}}^{(i)} = \{x_s^{(i)} \mid s \in \mathcal{I}(\mathcal{X})\}$. The exponential convergence of DC-ADMM assumes that all the agents in the network are connected. This connectivity assumption in the DC-ADMM is replaced with the variable connectivity assumption (see Assumption 1) in the LC-ADMM, which required that variable connectivity is satisfied per variable.

Finally, we remark on some properties of LC-ADMM. First, the LC-ADMM converges to the optimal solution even when considering other convex loss functions such as the Huber norm. This is a contrast to other distributed algorithm [6] whose derivation explicitly assumes specific noise distributions. Moreover, in the LC-ADMM, all the agents can update in parallel and the agents do not need global network topology information such as graph coloring. This is an especially desirable property for ad hoc networks. The LC-ADMM preserves *privacy* in the sense that each agent only needs to share the coupled variables with the neighbors who need that information. Agents only need to reveal the augmented factor information to their neighbors.

B. Large-Scale Networked Dynamical System

We revisit the estimation problem of large-scale networked dynamical systems (D1) and consider the convergence guarantees that are further specialized for these systems. To facilitate the discussion pertaining to observability, let us assume that the noise distribution is given by independent zero-mean Gaussian distributions and that a_t and c_t are twice differentiable and their gradients are Lipschitz continuous. Let $r_\tau \triangleq [r_{1,\tau}; \ldots, r_{|\mathcal{A}|,\tau}]$ be the state vector of the network at τ and $\mathcal{X} = [r_0; \ldots; r_t]$ be the trajectory of the network state up to time t. The linearized observability of (D1) at some reference trajectory X can be

analyzed using the following linearized observability matrix:

$$O(\mathcal{X}) = \begin{bmatrix} \partial c_0(r_0) \\ \partial c_1(r_1)\partial a_1(r_0) \\ \vdots \\ \partial c_t(r_t)\partial a_t(r_{t-1})\cdots\partial a_1(r_0) \end{bmatrix}. \quad (39)$$

For this example, we have the following corollary.

Corollary 1: Suppose that the dynamics model a_i and the measurement model c_i of (D1) are twice differentiable and their gradients are Lipschitz continuous. Also, suppose that the noise models are given by $w_{i,t} \sim \mathcal{N}(0, W_i^{-1})$ and $v_i \sim \mathcal{N}(0, V_i^{-1})$, where $W_i, V_i \succ 0$. Suppose that an optimal solution to (P2a) is given by $X_* = [r_{0*}; \ldots; r_{T*}]$ and the matrix $O(\mathcal{X}*)$ has a full column rank for at $\mathcal{X}_*$. Then, the objective $F(X)$ is locally strongly convex $\forall X \in \mathcal{B}_\rho(X_*)$ with some $\rho > 0$. Moreover, any trajectories starting within $\mathcal{B}_\rho(X_*)$ exponentially converge to X_*.

Proof: The proof is given in the Appendix. ■

An important implication of Corollary 1 is that the LC-ADMM only requires that the network as a whole is observable. Each agent, when considered in isolation from the other agents, need not be observable for the LC-ADMM to work. We validate this point later in an experiment in Section V-A. In addition, if (D1) is a linear time-varying (LTV) system with Gaussian noise, the exponential convergence is guaranteed globally.

Corollary 2 (Convergence for LTV Systems): In addition to the assumptions in Corollary 1, further assume that the dynamical system in (D1) is LTV. Then, the LC-ADMM is globally exponentially convergent to the unique solution. Moreover, each subproblem in (8) simplifies to an unconstrained quadratic program (QP), for which the analytical solution is given by the matrix multiplication of form $\hat{\mathcal{X}}_{k+1}^{(i)} = (H^{(i)\top}H^{(i)})^{-1}H^{(i)\top}b_k$. The matrix $(H^{(i)\top}H^{(i)})^{-1}H^{(i)\top}$ is only computed once at the beginning of LC-ADMM on agent i, and b_k is an affine function of Z_k and W_k.

Proof: The proof is similar to that of Corollary 1. The observability implies that $F(X)$ is strongly convexity and $F(X)$ is Lipschitz continuous subgradient. By Theorem 3, we have that the LC-ADMM globally converges exponentially to the global minimum. ■

Corollary 2 shows that each computation of LC-ADMM is quite inexpensive for linear systems once $(H^{(i)\top}H^{(i)})^{-1}H^{(i)\top}$ is computed for each agent i.

C. LC-ADMM for Problems With Weaker Assumptions

We consider the application of LC-ADMM with weaker assumptions. First, consider a case where the original problem (P2) additionally involves local inequality constraints $\hat{\mathcal{X}}^{(i)} \in S^{(i)}$, where $S^{(i)}$ is a convex set. In this case, one can modify the objective function (P2) to also include the indicator function

$$f_i(\hat{\mathcal{X}}^{(i)}) \leftarrow f_i(\hat{\mathcal{X}}^{(i)}) + \mathbb{I}_{S^{(i)}}(\hat{\mathcal{X}}^{(i)}).$$

Then, the LC-ADMM iterations in (8) and (9) can be written as

$$\hat{\mathcal{X}}_{k+1}^{(i)} = \underset{\hat{\mathcal{X}}^{(i)} \in S^{(i)}}{\arg\min}\ f_i(\hat{\mathcal{X}}^{(i)}) + \sum_{j\in\mathcal{N}^i}\sum_{s\in\mathcal{I}^{ij}} \frac{\beta}{2}\|x_s^{(i)} - \bar{x}_{s,k+1}^{(ij)} + \frac{1}{\beta}w_{s,k}^{(ij,i)}\|^2 \quad (40)$$

$$w_{s,k+1}^{(ij,i)} = w_{s,k}^{(ij,i)} + \beta\left(x_{s,k+1}^{(i)} - \bar{x}_{s,k+1}^{(ij)}\right). \quad (41)$$

Even though the additional inequality constraints can model a broader class of problems, the exponential convergence proof for Theorem 3 would not work anymore, so it was not included in the problem definition (P2a).

Next, we consider nonconvex optimization problems such as PGO for which Theorems 2 and 3 do not hold. While prior works in the literature showed that the ADMM converges for a certain class of nonconvex problems [38], (P2) does not satisfy all the assumptions of [38]. Therefore, it remains a future work to determine whether the LC-ADMM converges for a certain class of nonconvex problems. Instead, in this article, we demonstrate that the LC-ADMM can still be derived for nonconvex problems and empirically show that the LC-ADMM converges in some simulation examples.

We consider the problems involving manifold-constrained variables such as PGO. Suppose that a pose state in the d-dimensional space is given by $T = [R, t] \in \mathrm{SE}(d)$, where $R \in \mathrm{SO}(d)$ is the rotation component and $t \in \mathbb{R}^d$ is the translation component of T. If $T_p, T_q \in \mathcal{X}$ are the variables involving some factor $\phi \in \mathcal{F}$ with noisy relative pose observation $\tilde{T}_{pq}$, we can write the local consensus reformulation of the PGO problem as

$$\begin{aligned} \min_{X,Z} \quad & \sum_{i\in\mathcal{A}}\sum_{\phi\in\mathcal{F}_i} L(T_p^{(i)}, T_q^{(i)}, \tilde{T}_{pq}) \\ \text{subject to} \quad & \begin{matrix} T_s^{(i)} = T_s^{(ij)} \\ T_s^{(j)} = T_s^{(ij)} \end{matrix} \quad \forall s \in \mathcal{I}(\hat{\mathcal{X}}^{ij}), (i,j) \in \mathcal{E} \\ & T_s^{(i)} \in SE(d) \quad \forall s \in \mathcal{I}(\mathcal{X}^{(i)}), i \in \mathcal{A} \end{aligned} \quad (42)$$

where $L(T_p, T_q, \tilde{T}_{pq})$ represents some loss function on a relative pose. Like we did in (40), one can simply incorporate the manifold constraint in the local FGO update equation. The manifold constraint $T_s \in \mathrm{SE}(d)$ introduces the nonconvex constraints.

In practice, there are some heuristic strategies to mitigate the risks of being stuck at a local minimum. First, the local FGO update step of LC-ADMM does not preclude the use of optimization techniques that guarantee convergence to the global minimum for PGO (i.e., SE-Sync [11]). Therefore, loop-closure constraints within the agent itself are remedied. Second, we develop iDFGO, an incremental version of LC-ADMM, in Section IV where agents build the map incrementally. This incremental approach, combined with the ability to resolve the loop closures with self, makes the algorithm more robust against local minima. We also demonstrate empirically in simulations that LC-ADMM and iDFGO converge quickly to the optimal solution in Section V-D.

IV. Incremental DFGO

In the previous sections, we applied LC-ADMM to the multiagent MAP problem (1) over some fixed time horizon. If we take the system in (D1) as an example, the LC-ADMM estimates

$\mathcal{X}_t = \{r_{i,\tau} \mid i \in \mathcal{A}, \tau \in \mathcal{T}(t)\}$, where $\mathcal{T}(t) \triangleq [0, t]$ is the time horizon at t. In real-time applications, however, the distributed optimization problem needs to be recomputed at each time step t as new factors are added to the graph. As the length of the time horizon increases, the size of the problem in LC-ADMM also grows.

To address this challenge, we further extend LC-ADMM and derive the iDFGO algorithm that is scalable with respect to the increasing time horizon. The main idea is to combine a single-agent incremental probabilistic inference algorithm (i.e., iSAM2 [14]) with the local FGO update step of LC-ADMM such that only a recent subset of the factor graph needs to be recomputed. It turns out that simply applying an incremental algorithm to solve the local FGO problem does not result in an algorithm that is scalable with respect to time. To derive a time-scalable algorithm, we exploit the temporal sparsity of the real-time factor graph and the convergence of the augmented factors of LC-ADMM. The computation and communication bandwidth of the overall algorithm is both scalable in size of the network and in time.

For this section, we also additionally assume Gaussian distributions as they do in the derivation of iSAM2 [6], [14]. For problems with non-Gaussian distributions, one may consider using the moving time horizon approach where the width of the time horizon is kept constant.

A. Fluid Augmented Factor Update in Bayes Tree

One of the most prominent works for incremental probabilistic inference for a single robot is iSAM2 [14] which reformulates factor graphs as Bayes trees. The primary benefit of reformulating the original factor graphs as Bayes trees is that it permits an incremental update of the FGO solution. In the Bayes tree, the joint probability of the original factor graph is refactored as the product of conditional probabilities $\prod p(\delta_F|\delta_S)$ that is modeled as a chordal tree [14]. Each node is a conditional probability $p(\delta_F|\delta_S)$ for a clique of the original factor graph, and one can obtain this set of conditional probabilities by eliminating one variable at a time. Suppose that the factor f_{joint} depends on δ_F and δ_S, where δ_S is the separator variables and δ_F is the frontal variables

$$f_{\text{joint}}(\delta_F, \delta_S) \propto \exp\left(-\frac{1}{2}\|A_F\delta_F + A_S\delta_S - b\|^2\right). \tag{43}$$

The separator variables are those shared with its parent node, and they are eliminated from the node [14]. The probability distribution can be written as $f_{\text{joint}}(\delta_F, \delta_S) = P(\delta_F|\delta_S) f_{\text{new}}(\delta_S)$, where

$$P(\delta_F|\delta_S) \propto \exp\left(-\frac{1}{2}\|\delta_F + R\delta_S - d\|^2\right) \tag{44}$$

$$f_{\text{new}}(\delta_S) \propto \exp\left(-\frac{1}{2}\|A'\delta_S - b'\|^2\right). \tag{45}$$

Here, we have $R = A_F^\dagger A_S$, $d = A_F^\dagger b$, $A' = A_S - A_F R$, and $b' = b - A_F d$, and $A_F^\dagger = (A_F^\top A_F)^{-1} A_F^\top$ is the pseudoinverse of A_F. Finally, after the elimination, we have parent factor $f_{\text{new}}(\delta_S)$ and child factor $P(\delta_F|\delta_S)$. This elimination process is repeatedly applied to the top portion of the Bayes tree until only one parent factor remains as the root of the tree.

In iSAM2, the A' and b' terms for the parent are recomputed only when the linearization point changes sufficiently. Therefore, so long as the linearization point does not change too much, only the root of the tree needs to be modified in an incremental fashion when new factors are added. One of the assumptions that iSAM2 has is that the measurement values of the nonlinear factors (i.e., b term) are fixed in their lifetime. While this assumption holds true in the nominal single-agent FGO scenario, in which each measurement is observed once and does not change, we need to revisit the assumption for iDFGO.

Since iSAM2 scales well with respect to time, we extend it to distributed estimation problems. One naive attempt would be to simply solve the local FGO update step of LC-ADMM using iSAM2; however, this does not immediately result in a scalable algorithm. Because each augmented factor modifies its pseudo measurement at each LC-ADMM iteration, the assumption that the measurement term is fixed does not hold anymore. To compute the correct optimal MAP solution using the Bayes tree, every time b changes in a clique, the value of b' term in its parent node must be modified. This recursively affects the ancestors of the node, all the way up to the root of the tree. If there is an augmented factor near the bottom of the tree and it modifies its pseudo measurement, a large number of variables are affected in each update. Therefore, simply applying iSAM2 to the update step of LC-ADMM does not result in an algorithm that scales well with time.

We exploit the temporal structure of the real-time problem to address this issue. To this end, we make an important observation on the convergence of consensus errors. Recall that in the LC-ADMM, the role of the augmented factors is to enforce the consistency between a pair of variable estimates $x_s^{(i)}$ and $x_s^{(j)}$ for a variable x_s and agents i and j. In real-time applications, the effects of newer factors on older variables $x_s^{(i)}$ and $x_s^{(j)}$ diminish in comparison to those of augmented factors, and the two estimates converge over time. Fig. 5 shows some sample trajectories of the consensus error $\|x_s^{(i)} - x_s^{(j)}\|$ as a function of time from an electrical power grid example in Section V-A. At each time step, new factors are added to the graph and $K = 30$ LC-ADMM iterations are applied in a batch. Even though new factors are added to the graph, the error between each pair of estimates converges to zero around $t = 30$ after the augmented factor was created in this example.

We exploit this convergence of consensus error and propose the *fluid augmented factor update* where each augmented factor pair terminates its update once the consensus is achieved to some threshold. The fluid augmented factor update is summarized in Algorithm 2. If older augmented factors that reached consensus do not modify their pseudo measurements, the Bayes tree update only needs to recompute the more recent subset of the augmented factors. Fig. 6 shows the number of variables recomputed in a multiagent PGO example described later in Section V-D. Even though the number of factors in the graph grows over time, the number of affected variables remains

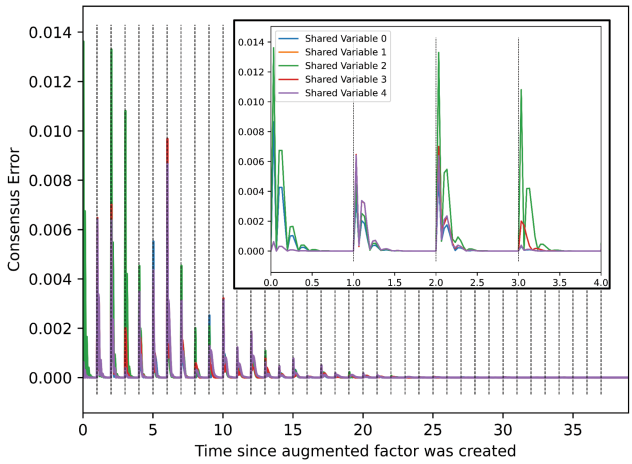

Fig. 5. Consensus error $\|x_s^{(i)} - x_s^{(j)}\|$ for five sample variables converges over time in the real-time electrical grid example (see Section V-A). New factors are added per time step (indicated by black dotted lines) and $K = 30$ LC-ADMM iterations are applied in batch at each time step. The consensus errors converge to zero over time.

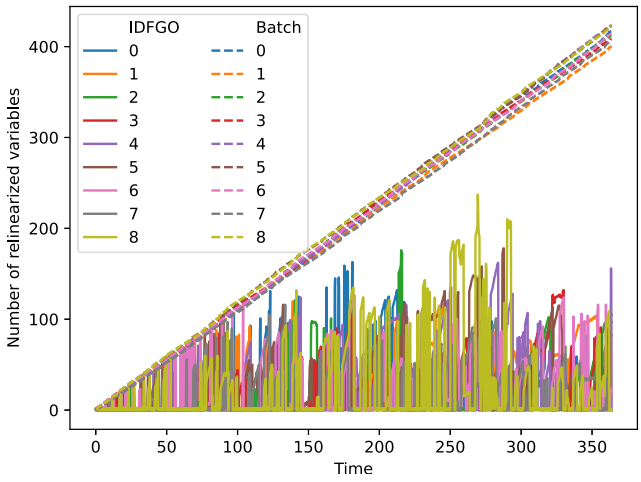

Fig. 6. Number of variables updated at each time step of iDFGO for the simulation example in Section V-D. As the size of the factor graph grows, the computational complexity of iDFGO remains bounded.

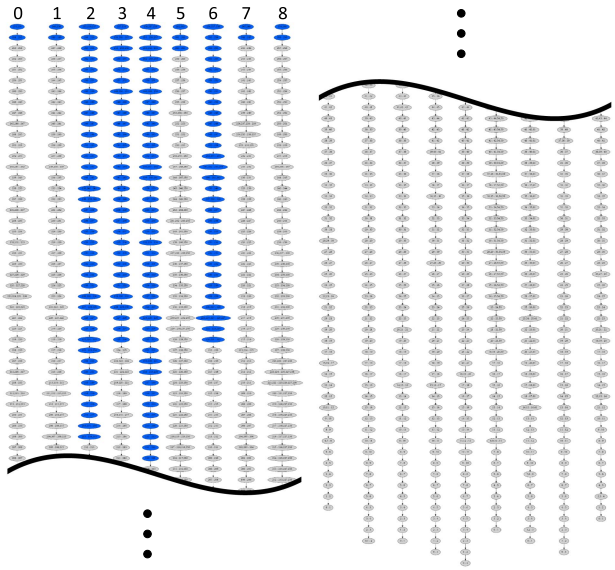

Fig. 7. Top of the Bayes trees (left) and the bottom (right) for the nine agents running iDFGO for real-time PGO example in Section V-D. The blue nodes show the variables whose nodes are modified by the incremental update.

Algorithm 2: Fluid Augmented Factor Update for Agent i, Neighbor j, Variable x_s, Time t, and Iteration k.

Result: Updated $\{w_{s,k,t}^{(ij,i)}\}$, $\{\bar{x}_{s,k,t}^{(ij)}\}$ and marked variable set M'

for *each* $s \in \mathcal{I}^{ij}$ *and* $j \in \mathcal{N}^i$ **do**

  if *not converged* **then**

    $\bar{x}_{s,k,t}^{(ij)} = \frac{1}{2}\left(x_{s,k,t}^{(i)} + x_{s,k,t}^{(j)}\right)$;

    $w_{s,k,t}^{(ij,i)} = w_{s,k-1,t}^{(ij,i)} + \frac{1}{\beta}\left(x_{s,k,t}^{(i)} - \bar{x}_{s,k,t}^{(ij)}\right)$;

    Update the pseudo measurement to $x_{s,k,t}^{(i)} - \frac{1}{\beta} w_{s,k,t}^{(ij,i)}$;

    Add variable $x_s^{(i)}$ to the set M';

  end

  else

    $\bar{x}_{s,k,t}^{(ij)} = \bar{x}_{s,k-1,t}^{(ij)}$;

    $w_{s,k,t}^{(ij,i)} = w_{s,k-1,t}^{(ij,i)}$;

  end

end

bounded in iDFGO. In addition to reducing the computational effort, fluid augmented factor update also improves the communication bandwidth. While the pair of augmented factors is converged, the neighboring robots do not need to continuously exchange the pseudo measurements for those factors. Therefore, fluid augmented factor update improves computation and communication simultaneously.

For the augmented factors that have not converged yet, the change in the value of pseudo measurements must be reflected recursively up to the root of the Bayes tree. We implement this similarly to the *fluid relinearization* step of iSAM2. We mark all the affected cliques and their ancestors in the tree and redo their variable elimination. Compared to the single-agent iSAM2 case, the number of affected variables during redoing the top of the Bayes tree is typically larger because some augmented factors from the past have not yet converged. However, because these factors are involved in the cliques near the top of the Bayes tree, the number of variables involved in the update step is still reduced compared to optimizing the whole factor graph in a batch.

B. Variable Ordering

An important aspect to consider in developing the Bayes tree is the order of variable elimination. The iSAM2 algorithm uses the constrained column approximate minimum degree (constrained COLAMD) algorithm to compute the order of elimination such

that it reduces the sparsity fill-ins in the presence of loop closures. This heuristic algorithm works well in improving computational efficiency for single-agent mapping problems in the presence of loop closures. However, it actually leads to a large number of recomputed variables when used in conjunction with the augmented factor updates of iDFGO. Instead, we choose a temporal ordering for variable elimination, where the newer variables are placed closer to the root of the tree. This approach preserves the temporal structure in the Bayes tree, and therefore, the fluid augmented factor update only affects the top part of the tree. Fig. 7 shows a set of example Bayes trees obtained by this variable ordering scheme in a multiagent PGO example where the highlighted nodes are the cliques that are affected during the particular update.

C. iDFGO Algorithm

The overall algorithm for iDFGO is summarized in Algorithm 3. The local FGO update part of iDFGO can be seen as an extension of iSAM2, and some of its subroutines are reused. First, at each time step, new factors and variables are added to the local factors. Second, the fluid augmented factor update step modifies some of the pseudo measurement of augmented factors, as described in Algorithm 2. All the variables related to the updated augmented factors are marked and stored in a set M in this step. Next, if there are new augmented factors at any k, they are also added to the factor graph and the involved variables are also added to the set M. The fluid relinearization step of iSAM2 (see [14, Algorithm 5]) may relinearize some of the nonlinear factors and returns its own set of marked keys. These marked keys are also added to the set M. Afterward, we redo the top of the Bayes tree in a similar way as [14, Algorithm 6] with exception of two modifications. The first modification is that we use the modified marked key M, which contains the variables affected by both fluid relinearization and fluid augmented factor update. We use this set to mark all the affected cliques in the Bayes tree and their ancestors. The second modification is the variable elimination order that we described in Section IV-B. After redoing the Bayes tree, neighboring agents locally exchange the new estimate. The iDFGO iterates this for K steps per each time step t. To promote an efficient implementation, the iDFGO algorithm is implemented in C++ using iSAM2 in the GTSAM library. We refer readers to [14] for details of the iSAM2 algorithm.

Algorithm 3: Incremental DFGO.

Result: $\hat{\mathcal{X}}_{t,k}^{(i)},\ i \in \mathcal{A}$
Each agent $i \in \mathcal{A}$ locally initializes $\hat{\mathcal{X}}_{0,K}^{(i)} = \varnothing$ and $t = 0$;
while *true* **do**
  Add any new factors;
  Initialize any new variables and add them to $\hat{\mathcal{X}}_{t,0}^{(i)}$;
  for *k=[0,K-1]* **do**
    Fluid augmented factor update (Alg. 2) yields marked variables M;
    Create any new augmented factors and add their variables to set M;
    Apply fluid relinearization (Alg. 5 in [14]). The marked variables from Alg 5 are also added to M;
    Taking M as input, redo the top of Bayes tree (Alg. 6 of [14]) that is modified to eliminate variables in a temporal ordering;
    Solve for delta Δ (Alg. 7 in [14]);
    The current estimate is given by $\hat{\mathcal{X}}_{t,k+1}^{(i)} = \hat{\mathcal{X}}_{t,k}^{(i)} \oplus \Delta$;
    Exchange shared variable estimates with neighbors;
  end
  $t \to t + 1$;
end

D. Scalability With Time

We discuss the complexity of the iDFGO algorithm with respect to the size of the trajectory. First, for an estimation of a dynamical system [e.g., system considered in (D1)], only the recent subset of nodes in the Bayes tree is updated instead of solving the optimization problem over the whole trajectory in a batch (see Fig. 7). The number of variables involved in the update has the complexity of $o(1)$ with respect to time.

In the presence of loop closures, the computational effort of the iDFGO algorithm adapts as necessary. Suppose that a newly added loop closure constraint affects variables from τ' time steps ago in the past. The variables directly involved in the loop closure (up to $t - \tau'$) as well as variables in its proximity are affected by the loop closure. If the change in the estimate is sufficiently large, the variables may have a large consensus error compared to the neighbors and may be required to apply the augmented factor update. The estimates of the distributed agents will converge again sometime after the loop closure is added, but the augmented factor continues to update until the consensus error converges sufficiently. In terms of algorithm complexity, suppose that there is an upper bound $T' > 0$ to the time lag on loop closure (e.g., $\tau' \leq T'$ for all loop closure). Then, the most general bound on the number of variables involved in each iDFGO update scales like $o(T'^3)$, and it is constant complexity with respect to the length of the whole trajectory.

V. Numerical Evaluation

We evaluate various aspects of LC-ADMM and iDFGO on multiple networked-system models using numerical simulations. We validate scalability, optimality, and convergence (see Section V-A) as well as distributed outlier rejection (see Section V-B). Finally, we compare the performance of LC-ADMM and iDFGO against other PGO algorithms (see Sections V-C and V-D). We use the Levenberg–Marquardt algorithm to solve each agent's individual FGO update step of the LC-ADMM, and we implement this using the GTSAM library in C++. The iDFGO algorithm is implemented using a modified version of iSAM2 [14], as described in Algorithm 3. The ground truth simulation and iDFGO are run on a single computer, and the

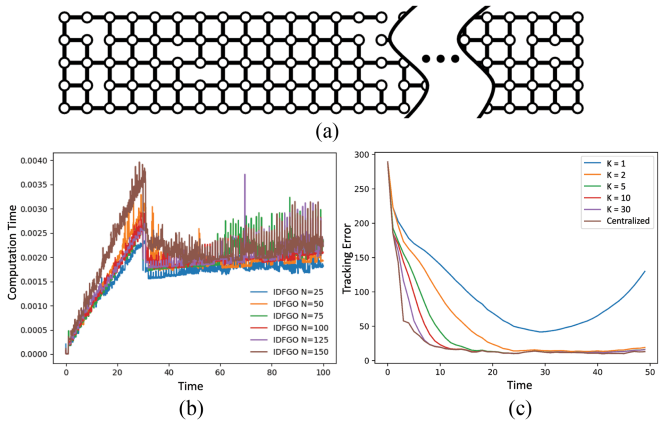

Fig. 8. Electrical power grid example. Dynamics is given by (46). (a) Network size N is varied by changing the number of columns in the grid. (b) Average computation time per node per iteration as a function of time. Each color shows one simulation with a different network size. As the network size increases, the local computational effort remains the same. (c) Optimality gap of iDFGO at steady state versus the number of LC-ADMM iterations K.

interagent communication is simulated by providing each agent with only its neighbors' information. All the timing results are obtained on a laptop with Intel 2.5-GHz i9-11900H processor.

A. Scalability, Convergence, and Optimality

Consider an example of a large-scale networked dynamical system (D1) given as a network of electrical power grids [9] involving hundreds of agents. Each plant is tasked to monitor its own state given the local measurement and the information exchanged with the neighboring plants. The dynamics and measurement models for each agent $i \in \mathcal{A}$ are given by

$$r_{i,t+1} = \sum_{j \in \mathcal{N}^i} A_{ij} r_{j,t} + w_{i,t}, \quad y_{i,t} = C_i r_{i,t} + v_{i,t} \tag{46}$$

$$A_{ii} = \begin{bmatrix} 1 & \Delta t \\ -\frac{k_i}{m_i}\Delta t & 1 - \frac{d_i}{m_i}\Delta t \end{bmatrix}, \quad A_{ij} = \begin{bmatrix} 0 & 0 \\ \frac{k_{ij}}{m_i}\Delta t & 0 \end{bmatrix},$$
$$C_i = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

The dynamics are locally coupled by A_{ij} terms, and the noise terms $w_{i,t}$ and $v_{i,t}$ are sampled from their respective independent identically distributed zero-mean Gaussian distributions. The system parameters k_{ij}, d_i, m_i^{-1} and Δt are set to 0.2, and $k_i = \sum_{(i,j)\in\mathcal{E}} k_{ij}$. The grid network has R rows and C columns with $N = R \times C$ agents in total, as shown in Fig. 8(a). The static network topology is defined such that each pair of adjacent agents has a 70% chance of having an edge between them. Because this system is linear, time-invariant, and observable, the convergence of LC-ADMM is globally exponential by Theorem 3 and Corollary 2.

We study the convergence of the consensus errors of LC-ADMM in Fig. 5. To isolate the effect of variablewise consensus, for Fig. 5 and this paragraph only, we apply the LC-ADMM in batch at each time step instead of iDFGO. Five samples of agent state $r_{i,\tau}$ are randomly selected, and the consensus error between $r^{(i)}_{i,\tau,k}$ and $r^{(j)}_{i,\tau,k}$ for a pair of neighbors i and j is tracked over each consensus iteration $k = [0, \ldots, K-1]$ and each time step since τ (i.e., $t = [\tau, \tau+1, \tau+2, \ldots]$). Each time step is indicated by the black vertical lines in Fig. 5, while there are $K = 30$ consensus iterations in between each time step. The new factors are added to the local factor graph at every time step. The figure shows that the consensus error converges over k between each time update, but the error increases each time the new factors are added to the graph. This is because the local estimate (i.e., $r^{(i)}_{i,\tau,k}$) changes based on the new information obtained from the augmented factor. Fig. 5 also shows that the magnitude of the error jumps at each time step decays over time. This is because $t - \tau$ is increasing and the effects of new factors on variable $r_{i,\tau}$ diminish over time. For all the variables sampled, neighbors reach a steady agreement (i.e., $r^{(i)}_{i,\tau,k} = r^{(j)}_{i,\tau,k}$) within 30 time steps since τ.

Next, we verify the scalability of iDFGO in network size and in time. The size of the grid network is varied from $N = 25$ to $N = 150$ agents, as shown in Fig. 8(a). For each simulation, $K = 30$ iterations are applied per time step. Fig. 8(b) shows the network-averaged computation time per LC-ADMM update as a function of time in a real-time scenario. The sharp drop in computation time near $t \approx 30$ is due to the internal mechanism within the Bayes tree update of the iSAM2 where it switches from calling a batch update to an incremental update. This switch happens because the ratio between the number of variables affected and the total number of variables in the problem goes under a threshold value. After $t \approx 30$, the computation time remains mostly constant even as the network size increases due to the locality property of LC-ADMM. We observe that the average computation time is similar among various network sizes. These plots confirm the scalability of iDFGO in both the size of the network and time.

Finally, we study the tradeoff between the convergence error and communication bandwidth for iDFGO. We run iDFGO for various numbers of ADMM iterations per time step K, where larger K means higher bandwidth. Suppose that a ground truth state of agent i time step t is denoted as $r_{i,t}$, and the real-time estimate by agent i is denoted as $r^{(i)}_{i,\tau,K}$, where $\tau = t$ and K is the total number of consensus iteration per time step. Then, the tracking error is defined as $\sum_{i\in\mathcal{A}} \|r^{(i)}_{i,\tau,K} - r_{i,t}\|$. For comparison, we also compute the centralized optimal estimation solution using a batch optimization method at each time horizon. This solution serves as the best possible error that the iDFGO algorithm could get. The results are shown in Fig. 8(c). The figure shows that the iDFGO tracks the true trajectory $K = 2, 5, 10, 30$, while the algorithm diverges for $K = 1$ because not enough consensus is applied at each time step. The more frequent the communication is, the faster the algorithm converges to the centralized optimal solution. Over time, however, the iDFGO approximates the centralized optimal solution even with a relatively small number of communication per time step.

B. Distributed Outlier Rejection on Locally Coupled System

Next, we demonstrate that non-Gaussian noise models can be used with the LC-ADMM to solve distributed outlier rejection problems. Continuing with the power grid example, suppose

now that the observation model in (46) is further corrupted by some sparse outlier noise

$$y_{i,t} = C_i r_{i,t} + v_{i,t} + \psi_{i,t}$$

where $\psi_{i,t}$ is a sparse but large noise. This type of outlier is particularly relevant for large-scale networks that are low-cost but have a higher risk of failure and makes accurate estimation more challenging. Unfortunately, algorithms that explicitly assume Gaussian distribution perform poorly under spurious outliers.

Our approach is to alternatively use convex loss functions that correspond to probability distributions with fatter tails. This modification can be incorporated naturally in iDFGO since the derivation of LC-ADMM does not assume a specific type of noise model. One example of an alternative loss function that models a fatter tail is the Huber loss function [46], which is defined as

$$L_\alpha(x) \triangleq \begin{cases} x^2, & \text{for } |x| \leq \alpha \\ \alpha(2|x| - \alpha), & \text{otherwise} \end{cases} \tag{47}$$

for some $\alpha > 0$. The individual agent's objective using the Huber loss function can be written as

$$f_i(\hat{\mathcal{X}}^{(i)}) = \sum_t L_\alpha(\|r_{i,t} - A_i r_{\mathcal{N}^i,t-1}\|_{W_i^t}) + L_\alpha(\|y_{i,t} - C_i r_{\mathcal{N}^i,t}\|_{V_i^t}).$$

The resulting subproblem can be written in QP form for which efficient algorithms exist. Because the Huber loss is convex, $o(1/k)$ convergence by Theorem 2 holds.

The simulation considers 400 agents in a connected 20×20 grid. If the nominal observation noise has standard deviation σ, the sparse outlier noise is simulated by setting $\psi_i^t = \pm 20\sigma$ with 2.5% probability of occurrence, and otherwise, $\psi_i^t = 0$. Even though the chance of having each outlier is small, there are several outliers somewhere in the network at each time step. We compare the results of applying LC-ADMM with two different loss functions: a 2-norm loss function (baseline) and a Huber loss function (robust estimation). Since the iDFGO assumes Gaussian noise (like iSAM2 does), we apply the LC-ADMM over a sliding time horizon of width $T = 30$ for this simulation example only.

Fig. 9 shows the measurement errors with respect to the first state variable as well as the estimation errors for the baseline and robust estimation algorithms. The figure shows the tracking errors for nine randomly selected agents out of 400. The baseline algorithm is significantly impacted by the outlier-corrupted signal, while the robust estimation algorithm rejects the outlier well, maintaining low estimation error. To the best of the authors' knowledge, this is the first demonstration of the distributed and localized estimation algorithm that can explicitly reject sparse outliers in a unified framework.

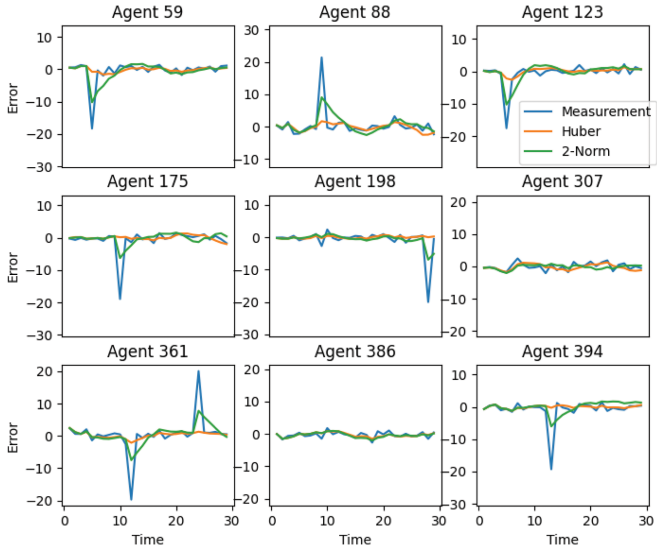

Fig. 9. Distributed outlier rejection in an power grid example using $\alpha = 1.35$ [46], $\beta = 1$, and K=10. The raw observation error (blue), robust estimate (yellow), and quadratic estimate (green) are shown for nine randomly selected agents out of 400. Errors are defined with respect to the ground truth from the simulation. The robust algorithm successfully rejects the spurious noises that otherwise affect the quadratic estimate.

C. Multiagent PGO via LC-ADMM

Next, we apply LC-ADMM and iDFGO to the multiagent PGO problems (42). This section evaluates LC-ADMM on a batch multiagent PGO, while the next section evaluates iDFGO on a real-time problem. We note that the LC-ADMM does not show a theoretical guarantee for convergence in nonconvex problems such as PGO. However, instead, we empirically show that the LC-ADMM locally converges to the centralized optimal solution in these examples. The batch PGO problem was validated on numerical simulations using a benchmark SLAM dataset Manhattan 3500 [47], split into five segments to emulate multiagent scenarios.

We compare the estimates computed by the LC-ADMM with two state-of-the-art PGO algorithms SE-Sync [11] and DC2-PGO [7]. SE-Sync is a centralized algorithm that is known to converge globally under mild conditions. We solve the centralized PGO problem using SE-Sync and refer to this solution as the centralized optimal solution. The DC2-PGO algorithm is a certifiably correct distributed PGO algorithm (for batch problems). The DC2-PGO algorithm is implemented for comparison against the LC-ADMM. We use the accelerated version of DC2-PGO, and the rank is initialized as $r = 2$. Note that DC2-PGO optimizes a loss function defined as $L(T_q, T_p, \tilde{T}_{pq}) =$

$$\kappa_{pq}\|R_q - R_p\tilde{R}_{pq}\|_F^2 + \tau_{pq}\|t_p - t_q - R_p\tilde{t}_{pq}\|^2 \tag{48}$$

for some $\kappa_{pq} > 0$, $\tau_{pq} > 0$, while the loss function for LC-ADMM is defined as $L(T_q, T_p, \tilde{T}_{pq}) =$

$$\kappa_{pq}\|\log\left(R_q^{-1}R_p\tilde{R}_{pq}\right)^\wedge\|^2 + \tau_{pq}\|t_p - t_q - R_p\tilde{t}_{pq}\|^2. \tag{49}$$

We denote $\log : \mathrm{SE}(d) \to \mathfrak{se}(d)$ as the logarithm map and $(\cdot)^\wedge : \mathbb{R}^{d\times d} \to \mathbb{R}^d$ is the inverse of the skew-symmetric matrix operator.

Once both estimates are computed using their respective loss functions, we use (48) to quantify the estimation error for evaluation purposes only. Both the algorithms are initialized with the optimal solution to the single-agent PGO using the observations

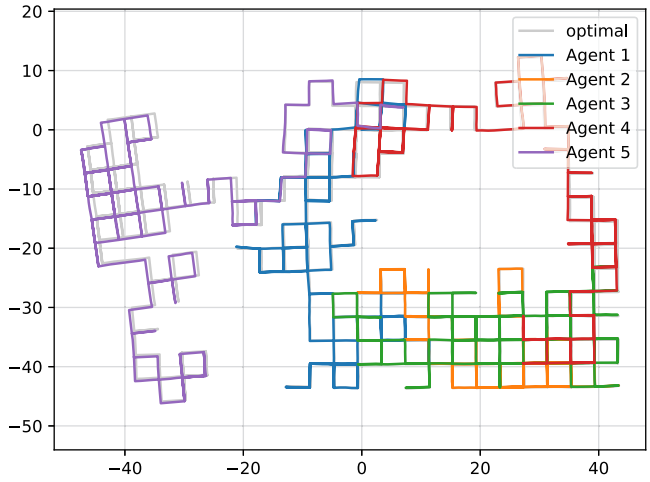

Fig. 10. Five agents cooperatively solve the distributed PGO problem. The pose estimates are shown for the LC-ADMM only after $k = 10$ iterations (colored) and for the optimal solution (gray) computed by a centralized method (SE-Sync [11]).

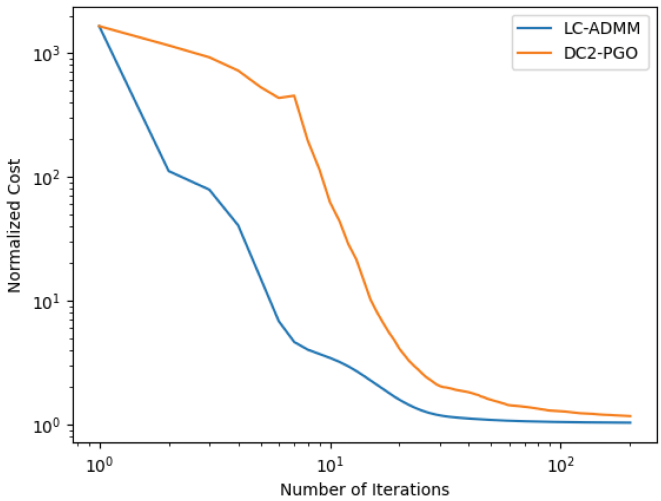

Fig. 11. Normalized cost versus number of iterations for LC-ADMM and DC2-PGO [7] on the M3500 dataset.

available only to itself (computed using a certifiably correct single-agent PGO algorithm [11]).

First, the colored trajectories in Fig. 10 show the LC-ADMM estimate by the five agents after $k = 10$ iterations. Only after a small number of iterations, the LC-ADMM converges to the centralized optimal solution (gray, computed by SE-Sync). The figure also qualitatively confirms that the LC-ADMM is not stuck at some local minima. Next, Fig. 11 shows the convergence rates of LC-ADMM and DC2-PGO.[4] Each agent only contributes its own poses and not a copy of the other agents' poses. The cost is defined as the centralized PGO objective function with (48) as the loss function and evaluated at $\tilde{\mathcal{X}}$. The optimal cost f^* is evaluated at the globally optimal solution to the centralized PGO computed by [11], and the normalized cost is defined as f/f^*. The normalized cost for both algorithms is shown as a function of iterations in Fig. 11. While the LC-ADMM does not have any guarantees to converge to the correct global optimal solution for a general nonconvex problem, Fig. 11 shows that the LC-ADMM converged to the optimal solution in this specific example. The LC-ADMM also converged faster than DC2-PGO even though both the algorithms start from the same initial trajectories. This result empirically suggests the applicability of LC-ADMM when provided with an initial guess that is good enough to avoid the local minimum.

[4]In a general multiagent PGO, the outer loop of DC2-PGO iterates on possible ranks of the low-rank semidefinite programming. In this example, DC2-PGO only iterates this outer loop once before convergence. Therefore, the convergence of DC2-PGO in Fig. 11 equivalently represents the convergence of its inner loop, the RBCD algorithm [7].

D. Multiagent PGO via iDFGO

Next, we evaluate iDFGO in a real-time multiagent PGO example. In this example, nine robots traverse an environment in a formation. The trajectories of the nine agents are shown on the left in Fig. 2. The task is to maintain both a good global map as well as good relative estimates. The dataset contains noisy observations for: 1) odometry; 2) relative pose between neighbor robots at a given time (shown in red); and 3) loop closures with some nearby poses (shown in blue). Without any interagent cooperation, the estimate that each agent computes will have a large drift as well as a large relative pose estimation error (shown in the center of Fig. 2). We set the number of consensus iterations applied at each time step to be $K = 5$.

A snapshot of the trajectory estimates computed by the iDFGO algorithm at time $t = 320, k = 0$ is shown on the right in Fig. 2. Each color corresponds to a different agent's trajectory estimate, and the gray plot shows the centralized optimal solution computed SE-Sync [11]. It shows that the trajectories estimated by iDFGO match the optimal solution. The darker color in each agent's trajectory denotes variables that were affected by the iDFGO update at step, while the lighter color denotes variables that were unaffected. The figure shows that iDFGO successfully updates the recent subset of the factor graph in an incremental fashion. Some of the agents have a larger number of affected variables than others. This is explained by the loop closure constraints on those agents affecting more variables from the past.

More details of the number of affected variables over the trajectory are shown in Fig. 6. The solid lines show the time history of the number of affected variables at the given time, while the dotted line shows the total number of variables updated if the whole trajectory of each agent is estimated in a batch. While the problem size increases for the batch case time increases, the size remains bounded for the iDFGO algorithm.

This simulation demonstrated that the iDFGO algorithm solves the multiagent PGO for a team of robots traversing an environment in a formation. The estimate matches the centralized optimal solution well. The algorithm is efficient as it involves a small number of consensus iterations per time step and the factor graphs are updated in an incremental fashion, leading to a more time-scalable algorithm.

VI. Conclusion

This article presented new algorithms to solve certain classes of the DFGO problems. The MAP estimation problem was formulated as a spatially decomposable FGO problem by exploiting the sparsity structure of locally coupled systems. First, we showed that the LC-ADMM algorithm has various advantages when solving DFGO in a batch over a fixed time horizon. The LC-ADMM scales well with the number of agents, incorporates robust convex loss objectives, solves problems fully in parallel, and does not need information about global graph topology. We have two new theoretical convergence results on the LC-ADMM: 1) it converges at $o(1/k)$ rate if the objective function is convex and has a unique solution and 2) it converges exponentially if the objective is strongly convex and its subdifferential is Lipschitz continuous. Next, using LC-ADMM as a backbone, we derived the iDFGO algorithm for real-time problems. The iDFGO algorithm integrates LC-ADMM with an extended version of iSAM2 to incrementally recompute the local FGO problem. The iDFGO is scalable both in the number of agents and in time.

Finally, the performance of LC-ADMM and iDFGO was evaluated in numerical simulations with examples from the electrical power grid and multiagent PGO. The simulation results verified scalability properties and illustrated the tradeoffs between optimality and communication. The multiagent PGO examples showed that the algorithm converged to the correct optimal solution despite problems being nonconvex, indicating the possible applicability to a broad class of problems.

Appendix A

Proof of Lemma 2: If F is convex, we have $\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle \geq 0$. Using (27)–(30), we may rewrite the left-hand side as . Substituting (27) into $\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle$, we get

$$\begin{aligned}
&\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle \\
&= \langle X_{k+1} - X_*, -\underline{M}^\top (Y_{k+1} - Y_*) - A_3^\top (V_{k+1} - V_*) \rangle \\
&\quad + \beta \langle X_{k+1} - X_*, \overline{M}^\top (Z_k - Z_{k+1}) \rangle \\
&= \frac{2}{\beta} \langle Y_k - Y_{k+1}, Y_{k+1} - Y_* \rangle + \frac{1}{\beta} \langle V_k - V_{k+1}, V_{k+1} - V_* \rangle \\
&\quad + 2\beta \langle Z_{k+1} - Z_*, Z_k - Z_{k+1} \rangle \\
&= 2(U_{k+1} - U_*)^\top G (U_k - U_{k+1}) \\
&= \|U_k - U_*\|_G^2 - \|U_{k+1} - U_*\|_G^2 - \|U_k - U_{k+1}\|_G^2.
\end{aligned}$$

This proves (31). Because F is convex, we have $\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle \geq 0$, and therefore, (31) holds. When F is strongly convex, we automatically have X_* (and thus Z_*) are unique. In addition, by Definition 3, we have

$$m_F \|X_{k+1} - X_*\|^2 \leq \langle X_{k+1} - X_*, g_{k+1} - g_* \rangle.$$

Therefore, (33) holds. ■

Proof of Lemma 4: We start by considering the following inequality, which holds true $\forall \mu > 0$ [39]:

$$\|a_1 + a_2\|^2 + (\mu - 1)\|a_1\|^2 \geq (1 - 1/\mu)\|a_2\|^2. \tag{50}$$

We select $\mu > 1$ so that all the terms are positive, and it results in a meaningful bound. If we let $a_1 = g_{k+1} - g_*$, $a_2 = \underline{M}^\top (Y_{k+1} - Y_*) + A_3^\top (V_{k+1} - V_*)$, we have $a_1 + a_2 = \beta \overline{M}^\top (Z_k - Z_{k+1})$ by (27). The left-hand side of (50) is upper-bounded by

$$\begin{aligned}
&\|\beta \overline{M}(Z_{k+1} - Z_k)\|^2 + (\mu - 1)\|g_{k+1} - g_*\|^2 \\
&\leq \beta^2 \sigma_{\max}^2(\overline{M}) \|Z_{k+1} - Z_k\|^2 + (\mu - 1) L_F^2 \|X_{k+1} - X_*\|^2
\end{aligned} \tag{51}$$

where we used (12) from Lipschitz continuous subdifferential assumption. Next, we have that $Y_k \in \mathrm{Im}(\underline{M})$ and $V_k \in \mathrm{Im}(A_3)$ $\forall k > 0$ by (22) and (21). Therefore, the right-hand side of (50) can be lower bound by

$$\begin{aligned}
&\|\underline{M}^\top (Y_{k+1} - Y_*) + A_3^\top (V_{k+1} - V_*)\|^2 \\
&= \left\| \begin{bmatrix} \underline{M} \\ A_3 \end{bmatrix}^\top \begin{bmatrix} Y_{k+1} - Y_* \\ V_{k+1} - V_* \end{bmatrix} \right\|^2 \\
&\geq \tilde{\sigma}_{\min}^2(M) \left\| \begin{bmatrix} Y_{k+1} - Y_* \\ V_{k+1} - V_* \end{bmatrix} \right\|^2 \\
&= \tilde{\sigma}_{\min}^2(M) (\|Y_{k+1} - Y_*\|^2 + \|V_{k+1} - V_*\|^2)
\end{aligned} \tag{52}$$

where $M \triangleq [\underline{M}; A_3]^\top$ and $\tilde{\sigma}_{\min}(M)$ is the minimum nonzero singular value of M. Therefore, we have

$$\begin{aligned}
&\beta^2 \sigma_{\max}^2(\overline{M}) \|Z_{k+1} - Z_k\|^2 + (\mu - 1) L_F^2 \|X_{k+1} - X_*\|^2 \\
&\geq \left(1 - \frac{1}{\mu}\right) \tilde{\sigma}_{\min}^2(\underline{M}) (\|Y_{k+1} - Y_*\|^2 + \|V_{k+1} - V_*\|^2).
\end{aligned}$$

This is further rearranged to

$$\begin{aligned}
&\beta C_1 \|Z_{k+1} - Z_k\|^2 + \frac{C_2}{\beta} \|X_{k+1} - X_*\|^2 \\
&\geq \frac{1}{\beta} \|Y_{k+1} - Y_*\|^2 + \frac{1}{\beta} \|V_{k+1} - V_*\|^2
\end{aligned}$$

where

$$C_1 = \frac{\mu \sigma_{\max}^2(\overline{M})}{(\mu - 1)\tilde{\sigma}_{\min}^2(\underline{M})}, \quad C_2 = \frac{\mu L_F^2}{\tilde{\sigma}_{\min}^2(\underline{M})}.$$

Using $\|Z_{k+1} - Z_*\| \leq \frac{1}{2}\sigma_{\max}(\overline{M}) \|X_{k+1} - X_*\|$, we get

$$\begin{aligned}
&\beta C_1 \|Z_{k+1} - Z_k\|^2 + \left(\frac{C_2}{\beta} + \frac{\beta}{4} \sigma_{\max}^2(\overline{M})\right) \|X_{k+1} - X_*\|^2 \\
&\geq \beta \|Z_{k+1} - Z_*\|^2 + \frac{1}{\beta} \|Y_{k+1} - Y_*\|^2 + \frac{1}{\beta} \|V_{k+1} - V_*\|^2.
\end{aligned}$$

Then, $c > 0$ as defined in (35) satisfies

$$\begin{aligned}
&\beta \|Z_{k+1} - Z_k\|^2 + m_F \|X_{k+1} - X_*\|^2 \\
&\geq c\beta \|Z_{k+1} - Z_*\|^2 + \frac{c}{\beta} \|Y_{k+1} - Y_*\|^2 + \frac{c}{\beta} \|V_{k+1} - V_*\|^2.
\end{aligned}$$

This further satisfies (36). ■

Proof of Corollary 1: The objective function in (P1) is given by

$$F(X) = \sum_{t=1}^{T} \|r_t - a_t(r_{t-1})\|^2_{W_t} + \sum_{t=0}^{T} \|y_t - c_t(r_t)\|^2_{V_t}$$
$$= R^\top(X)\Sigma R(X)$$

where $\Sigma = \text{diag}(W_1, \ldots, W_T, V_0, \ldots, V_T)$ and $R(X) = [r_1 - a_1(r_0); \ldots; r_T - a_T(r_{T-1}), y_0 - c_0(r_0), \ldots, y_T - c_T(r_T)]$. Since Σ is a positive-definite matrix, the Hessian of $F(X)$ is strictly positive definite at each $X \in \mathcal{B}_\rho(X_*)$ if and only if $\partial R(X)$ has a full column rank at X.

The residual Jacobian can be written as $\partial R(X)$ equals to

$$-\begin{bmatrix} \partial a_1(r_0) & I & 0 & \cdots & 0 \\ 0 & \partial a_2(r_1) & I & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \partial a_T(r_{T-1}) & I \\ \partial c_0(r_0) & 0 & \cdots & & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & & 0 & \partial c_T(r_T) \end{bmatrix}.$$

Applying block Gauss eliminations, one can write the first column as $[0; \ldots; 0; \partial c_0; \partial c_1 \partial a_1; \ldots; \partial c_T \partial a_T \cdots \partial a_1]$. Since X_* is a feasible solution to (P2a), there exists a solution $\mathcal{X}_*$ for (P1) such that $x^{(i)}_{s,*} = x_{s,*}$ for $\forall s \in \mathcal{I}(\mathcal{X}), \forall i \in \mathcal{A}$ by Lemma 1. Since the system is observable at $\mathcal{X}_*$, i.e., $O(\mathcal{X}_*)$, the first column of $\partial R(X_*)$ after the Gauss elimination has a full column rank. One can see that other block columns of $\partial R(X_*)$ are also linearly independent by noting their identity block elements. Therefore, $\partial R(X_*)$ has a full column rank, and the Hessian of $F(X_*)$ is positive definite. Since a_t and c_t are twice differentiable, there exists a local neighborhood $\mathcal{B}_\rho(X_*)$ with sufficiently small $\rho > 0$ in which $F(X)$ is strongly convex for $\forall X \in \mathcal{B}_\rho(X_*)$ with the local coefficient of strong convexity $m_F > 0$ is given by

$$m_F = \min_{X \in \mathcal{B}_\rho(X_*)} \sigma_{\min}(HF(X)) \tag{53}$$

where $\sigma_{\min}(\cdot)$ is the smallest eigenvalue of the matrix. Therefore, Assumption 3 is satisfied locally.

The objective $F(X)$ is also Lipschitz continuous subgradient, given that $a_t(r_{t-1})$ and $c_t(r_t)$ are Lipschitz continuous. This satisfies Assumption 2. Finally, we have that the LC-ADMM is locally exponentially stable at X^* by Theorem 3. ■

ACKNOWLEDGMENT

The authors would like to thank F. Y. Hadaegh, A. Rahmani, M. LaValle, M. Cua, and D. Zheng.

REFERENCES

[1] G. Han, H. Xu, T. Q. Duong, J. Jiang, and T. Hara, "Localization algorithms of wireless sensor networks: A survey," *Telecommun. Syst.*, vol. 52, no. 4, pp. 2419–2436, 2013.

[2] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi, "Collaborative sensor network localization: Algorithms and practical issues," *Proc. IEEE*, vol. 106, no. 6, pp. 1089–1114, Jun. 2018.

[3] F. Bullo, J. Cortés, and S. Martinez, *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[4] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, "GLAS: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE. Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4249–4256, Jul. 2020.

[5] K. Matsuka, A. O. Feldman, E. S. Lupu, S.-J. Chung, and F. Y. Hadaegh, "Decentralized formation pose estimation for spacecraft swarms," *Adv. Space Res.*, vol. 67, no. 11, pp. 3527–3545, 2021.

[6] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 5220–5227.

[7] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, "Distributed certifiably correct pose-graph optimization," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 2137–2156, Dec. 2021.

[8] U. A. Khan and J. M. Moura, "Distributing the Kalman filter for large-scale systems," *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 4919–4935, Oct. 2008.

[9] Y.-S. Wang, S. You, and N. Matni, "Localized distributed Kalman filters for large-scale systems," in *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 52–57, 2015.

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

[11] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special euclidean group," *Int. J. Robot. Res.*, vol. 38, nos. 2/3, pp. 95–125, 2019.

[12] T. Fan and T. Murphey, "Majorization minimization methods for distributed pose graph optimization with convergence guarantees," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5058–5065.

[13] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[14] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, 2012.

[15] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Boston, MA, USA: Now, 2011.

[16] R. Zhang and J. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1701–1709.

[17] K. Huang and N. D. Sidiropoulos, "Consensus-ADMM for general quadratically constrained quadratic programming," *IEEE Trans. Signal Process.*, vol. 64, no. 20, pp. 5297–5310, Oct. 2016.

[18] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *Proc. Amer. Control Conf.*, 2020, pp. 3619–3626.

[19] O. Shorinwa and M. Schwager, "Distributed contact-implicit trajectory optimization for collaborative manipulation," in *Proc. Int. Symp. Multi-Robot Multi-Agent Syst.*, 2021, pp. 56–65.

[20] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Distributed Kalman filtering based on consensus strategies," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 622–633, May 2008.

[21] R. Olfati-Saber, "Kalman-consensus filter: Optimality, stability, and performance," in *Proc. IEEE 48th Conf. Decis. Control/28th Chin. Control Conf.*, 2009, pp. 7036–7042.

[22] S. Park and N. C. Martins, "Design of distributed LTI observers for state omniscience," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 561–576, Feb. 2016.

[23] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, "Information weighted consensus filters and their application in distributed camera networks," *IEEE Trans. Autom. Control*, vol. 58, no. 12, pp. 3112–3125, Dec. 2013.

[24] S. Bandyopadhyay and S.-J. Chung, "Distributed Bayesian filtering using logarithmic opinion pool for dynamic sensor networks," *Automatica*, vol. 97, pp. 7–17, 2018.

[25] J. Anderson, J. C. Doyle, S. H. Low, and N. Matni, "System level synthesis," *Annu. Rev. Control*, vol. 47, pp. 364–393, 2019.

[26] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, 2006.
[27] G. Sibley, L. Matthies, and G. Sukhatme, "A sliding window filter for incremental SLAM," in *Unifying Perspectives in Computational and Robot Vision*. New York, NY, USA: Springer, 2008, pp. 103–112.
[28] S. Williams, V. Indelman, M. Kaess, R. Roberts, J. J. Leonard, and F. Dellaert, "Concurrent filtering and smoothing: A parallel architecture for real-time navigation and full smoothing," *Int. J. Robot. Res.*, vol. 33, no. 12, pp. 1544–1568, 2014.
[29] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph SLAM," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1371–1385, Dec. 2014.
[30] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.
[31] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[32] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.
[33] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.
[34] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, Mar. 2015.
[35] I. Necoara, V. Nedelcu, and I. Dumitrache, "Parallel and distributed optimization methods for estimation and control in networks," *J. Process Control*, vol. 21, no. 5, pp. 756–766, 2011.
[36] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Trans. Autom. Control*, vol. 57, no. 1, pp. 151–164, Jan. 2011.
[37] B. Gharesifard and J. Cortés, "Distributed continuous-time convex optimization on weight-balanced digraphs," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 781–786, Mar. 2014.
[38] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, no. 1, pp. 29–63, 2019.
[39] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.
[40] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization—Part I: Algorithm and convergence analysis," *IEEE Trans. Signal Process.*, vol. 64, no. 12, pp. 3118–3130, Jun. 2016.
[41] O. Shorinwa, J. Yu, T. Halsted, A. Koufos, and M. Schwager, "Distributed multi-target tracking for autonomous vehicle fleets," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3495–3501.
[42] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in MPC and network flows," *IEEE Trans. Autom. Control*, vol. 60, no. 7, pp. 2004–2009, Jul. 2015.
[43] G. Shi, W. Hönig, X. Shi, Y. Yue, and S.-J. Chung, "Neural-Swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1063–1079, Apr. 2022.
[44] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G ^{2} o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.
[45] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, "Parallel multi-block ADMM with o(1/k) convergence," *J. Sci. Comput.*, vol. 71, no. 2, pp. 712–736, 2017.
[46] P. J. Huber, *Robust Statistics*, vol. 523. Hoboken, NJ, USA: Wiley, 2004.
[47] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates,," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 2262–2269.

Kai Matsuka was born in New York, USA, in 1993. He received the B.S. degree in aerospace engineering from the University of California, Los Angeles, CA, USA, in 2016, and the M.S. degree in space engineering in 2018 from the California Institute of Technology, Pasadena, CA, where he is currently working toward the Ph.D. degree in space engineering from the Graduate Aerospace Laboratories.

From 2016 to 2017, he was a Guidance and Control Analysis Engineer with the NASA Jet Propulsion Laboratory, Pasadena, where he did research on fuel-efficient control of formation flying for the earth science mission. He is currently a Graduate Research Assistant with the Graduate Aerospace Laboratories, California Institute of Technology. His current research interests include distributed estimation, distributed optimization, and multiagent systems.

Mr. Matsuka is a recipient of the National Science Foundation Graduate Research Fellowship. He is a Reviewer for IEEE TRANSACTIONS ON AUTOMATIC CONTROL and IEEE TRANSACTIONS ON ROBOTICS.

Soon-Jo Chung (Senior Member, IEEE) received the B.S. degree (*summa cum laude*) in aerospace engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 1998, and the S.M. degree in aeronautics and astronautics and the Sc.D. degree in estimation and control from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002 and 2007, respectively.

He is currently a Bren Professor of Control and Dynamical Systems and a Jet Propulsion Laboratory Senior Research Scientist with the California Institute of Technology, Pasadena, CA, USA. From 2009 to 2016, he was with the faculty of the University of Illinois at Urbana—Champaign (UIUC), Champaign, IL, USA. His research interests include spacecraft and aerial swarms and autonomous aerospace systems, and in particular, the theory and application of complex nonlinear dynamics, control, estimation, guidance, and navigation of autonomous space and air vehicles.

Dr. Chung was the recipient of the UIUC Engineering Deans Award for Excellence in Research, the Beckman Faculty Fellowship of the UIUC Center for Advanced Study, the U.S. Air Force Office of Scientific Research Young Investigator Award, the National Science Foundation Faculty Early Career Development Award, a 2020 Honorable Mention for IEEE ROBOTICS AND AUTOMATION LETTERS Best Paper Award, and three Best Conference Paper Awards from the IEEE and the American Institute of Aeronautics and Astronautics. He is an Associate Editor for IEEE TRANSACTIONS ON AUTOMATIC CONTROL and *Journal of Guidance, Control, and Dynamics*. He was an Associate Editor for IEEE TRANSACTIONS ON ROBOTICS. He was a Guest Editor for a Special Section on Aerial Swarm Robotics published in IEEE TRANSACTIONS ON ROBOTICS.