Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Scalable, Federated Gaussian Process Training for Decentralized Multi-Agent Systems

GEORGE P. KONTOUDIS[1], (Member, IEEE), and DANIEL J. STILWELL[2], (Senior Member, IEEE)
[1]Department of Mechanical Engineering, Colorado School of Mines, Golden, CO 80401 USA (e-mail: george.kontoudis@mines.edu)
[2]Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: stilwell@vt.edu)

Corresponding author: George P. Kontoudis (e-mail: george.kontoudis@mines.edu).

This work was supported by the Office of Naval Research via grants N00014-18-1-2627 and N00014-19-1-2194.

ABSTRACT Gaussian process (GP) training of kernel hyperparameters still remains a major challenge due to high computational complexity. The typical GP training method employs maximum likelihood estimation to solve an optimization problem that requires cubic computations for each iteration. In addition, GP training in multi-agent systems requires significant amount of inter-agent communication that typically involves sharing of local data. In this paper, we propose scalable optimization algorithms for federated training using Gaussian processes (GPs) in decentralized multi-agent systems. To decentralize the implementation of GP training with maximum likelihood estimation, we employ the alternating direction method of multipliers (ADMM). We provide a closed-form solution of the decentralized proximal ADMM for the case of GP hyperparameter training using the separable squared exponential kernel. Federated learning is promoted in decentralized networks by prohibiting local data exchange between agents. Moreover, we extend a centralized GP training method by augmenting local datasets to improve the GP training estimation accuracy with large-scale multi-agent systems. The efficiency of the proposed methods is illustrated with numerical experiments.

INDEX TERMS Gaussian processes, multi-agent systems, federated learning.

I. INTRODUCTION

TEAMS of agents have received considerable attention in recent years, as they can address tasks that cannot be performed efficiently by a single entity. Multi-agent systems are attractive for their inherent property of collecting simultaneously data from multiple locations—a group of agents can collect more data than a single agent during the same time period. Central to machine learning (ML) methodologies is the collection of large datasets in order to ensure reliable training. To this end, networks of agents favor learning techniques due to their data collection capabilities. However, they face major challenges including limited computational resources and communication restrictions. A typical approach to address these challenges relies on centralizing the collected data in a single node (e.g., cloud or data center), which requires high computational and storage resources. However, gathering data on a central server may cause network traffic congestion and security/privacy issues. To ensure data privacy, a promising solution is federated learning (FL) [1]. FL aims to implement ML techniques in centralized or decentralized networks, but without real data communication to comply with the General Data Protection Regulation (GDPR) of the EU / UK [2]. For certain applications, such as in GPS-denied environments, it is unfeasible to implement ML algorithms in a centralized network, as distant nodes may not be able to establish communication directly with the central node due to communication range limitations or bandwidth [3]. Yet, even if we manage to collect all data in a central node, the time and space complexity for rapid updates of the ML models require resources that are not available to agents operating in the field.

Our aim in this work is to develop fully decentralized algorithms for approximate Gaussian process (GP) training that relax the computation and communication requirements with no data sharing and achieve similar performance to centralized GP training methods. We propose the first decentralized methods to estimate GP hyperparameters with maximum likelihood estimation, based on the alternating direction method of multipliers (ADMM) [4]. The decentralized GP training concept is presented in Fig. 1. Let us consider an unknown spatial field, where each agent samples the environment and maintains a local dataset. The proposed decentralized GP training methods (middle row in Fig. 1) consist of three steps: i) train a local GP surrogate model based on the information of local datasets; ii) decentralized coordination of local GP models; and iii) decentralized aggregation of a global GP model. Step (ii) requires the solution of a distributed opti-

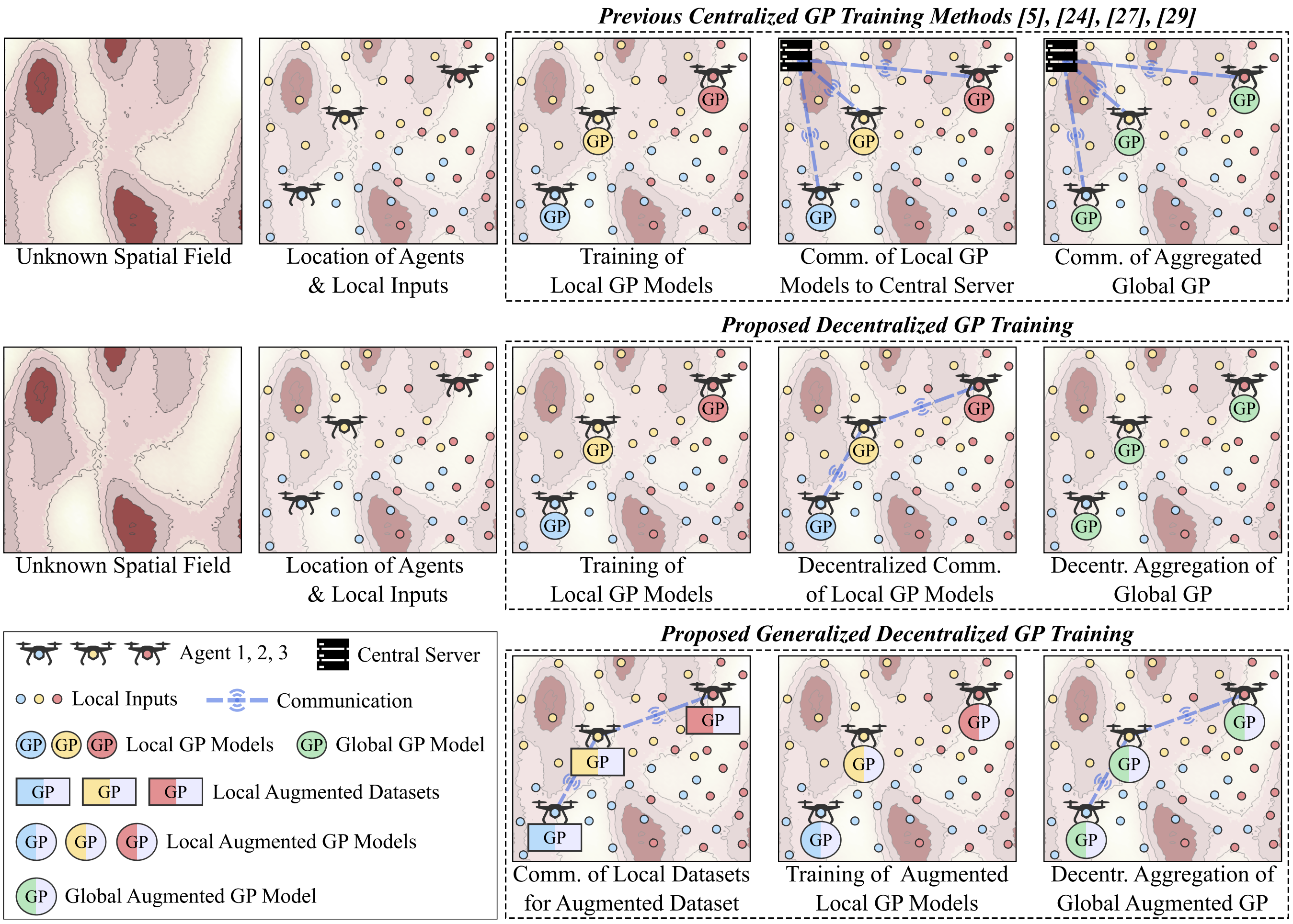
Previous Centralized GP Training Methods [5], [24], [27], [29]
Unknown Spatial Field
Location of Agents & Local Inputs
Training of Local GP Models
Comm. of Local GP Models to Central Server
Comm. of Aggregated Global GP
Proposed Decentralized GP Training
Decentralized Comm. of Local GP Models
Decentr. Aggregation of Global GP
Proposed Generalized Decentralized GP Training
Comm. of Local Datasets for Augmented Dataset
Training of Augmented Local GP Models
Decentr. Aggregation of Global Augmented GP
GP
Agent 1, 2, 3
Central Server
Local Inputs
Communication
Local GP Models
Global GP Model
Local Augmented Datasets
Local Augmented GP Models
Global Augmented GP Model

FIGURE 1. Decentralized GP training in multi-agent systems using federated learning. The exploration area of interest is an unknown spatial field. The unknown field can be of higher input dimension, i.e. $D > 2$, but for illustrative reasons we depict a 2D input environment. Same colored circles correspond to a local dataset of the matching colored agent. Each agent trains a local GP model using its dataset. The agents coordinate to produce a global GP model with no data exchange that promotes federated learning.

mization problem with ADMM. The proposed generalized decentralized GP training (bottom row in Fig. 1) requires three steps: i) communication of local datasets to generate the local augmented datasets; ii) train a local GP surrogate model based on local augmented datasets; and iii) decentralized coordination of local augmented GP models to produce the global augmented GP model. We also extend a centralized method [5] by augmenting local datasets to improve the GP hyperparameter estimation accuracy for the case of large fleet multi-agent systems.

Gaussian processes [6], [7] are used in various multi-agent applications [8]–[18], but their major disadvantage is the poor scalability with the number of observations. In particular, provided N observations, the training entails $\mathcal{O}(N^3)$ and the prediction requires $\mathcal{O}(N^2)$ computations. Although GP training is significantly more expensive than GP prediction, the majority of research is focused on improving the GP prediction scalability by assuming *a priori* knowledge of the hyperparameters. This is a strong assumption and in practice leads to inaccurate regression and deteriorates the adaptability of GPs. Another limitation for the implementation of GPs in multi-agent systems is the communication [3]. For centralized GPs, every agent has to communicate to a central node. However, excessive communication is challenging in decentralized networks, because the agents can pass messages only within a range [19], which may vary in space and time [20].

Two major research directions for GP approximations are based on global and local approaches [21]. Global approximation methods promote sparsity by using either a subset of N_{sub} observations or by introducing a set of N_{sub} pseudo-inputs, where $N_{\text{sub}} \ll N$ to perform GP training with a much smaller dataset [22], [23]. Sparse GPs have been used in mobile sensor networks to model spatial fields [9]. In [8], a GP with truncated observations is proposed, and in [10] a subset of observations is used for traffic modeling. These methods require global knowledge of the observations, which increases inter-agent communications. Also, the interpolation property is not retained with pseudo-input methods.

The second research direction uses local approximation methods to reduce the computational burden of GP training.

A local approximate method with maximum likelihood estimation (MLE) is the factorized GP training [24] (FACT-GP). That is a centralized algorithm which is based on a server-client structure. The main idea is to assume statistical independence between clients, which results in the approximation of the inverse covariance matrix by the inverse of a block diagonal matrix. To this end, a significant reduction in computing the inverse of multiple covariance matrices is achieved at the cost of excessive communication overhead. More specifically, every local entity transmits multiple inverted blocks of the covariance matrix per MLE iteration. Multiple studies revealed that the alternating direction method of multipliers (ADMM) [4] is appealing in centralized multi-agent missions [25], [26]. Xu *et al.* [27] reformulated the FACT-GP method by using the consensus ADMM [4] (c-GP). The c-GP method reduces the communication overhead of GP training, but requires high computational resources to solve a nested optimization problem at every ADMM-iteration. Subsequently, the authors in [5] employed the inexact proximal ADMM [28] to alleviate the computation demand (apx-GP). However, both ADMM-based factorized GP training methods (c-GP and apx-GP) require a centralized network topology and perform poorly with large fleet multi-agent systems. In [29], the authors introduced an efficient centralized methodology for large fleet multi-agent systems termed as generalized factorized GP training (g-FACT-GP). The latter entails additional communication between agents to enrich local datasets with a global random dataset that violates the requirements of federated learning.

Federated GP learning has recently received attention. In [30], the authors presented an module-driven GP method where each module is characterized by a local sparse GP model. Then, a global sparse GP model is trained with no data exchange between modules. The main idea is an extension of FACT-GP training [24], but with stochastic variational inference. The authors in [31] derived convergence bounds for a federated GP method that employs an aggregation strategy and stochastic gradient descent. This federated GP method has also been applied to multi-output GPs [32]. All these federated GP methods are distributed, but require a central entity for the aggregation. Our focus with federated GP training is on decentralized networks with inter-agent communication.

In [33] we present preliminary results of this work. In particular, we discuss the main concept of DEC-c-GP and DEC-apx-GP. This paper introduces a decentralized generalized GP training method (DEC-gapx-GP) that can be used in large-scale multi-agent systems. In addition, we discuss an improved centralized GP training method (gapx-GP) that extends [5] for large-scale multi-agent systems. Finally, we provide a rigorous proof for DEC-apx-GP in Appendix C, derive the computation, space, and communication complexity of all proposed methods (gapx-GP, DEC-c-GP, DEC-apx-GP, DEC-gapx-GP), and provide additional numerical examples that illustrate the efficiency of our methods.

The contribution of this paper is the formulation of decentralized GP training methods (DEC-c-GP, DEC-apx-GP, and DEC-gapx-GP). The proposed decentralized methods cover a broad spectrum of multi-agent missions for GP training and they simultaneously achieve similar GP model accuracy with centralized GP training methods [5], [24], [29] and global GPs [6]. The first method (DEC-c-GP) is computationally expensive, but provides accurate model estimation for small and medium fleet sizes. Next, we derive a closed-form solution for the GP hyperparameter optimization problem (DEC-apx-GP). The latter enables scalable computations with significantly faster GP training than global GPs and achieves accurate model estimation for small and medium fleet sizes. Both DEC-c-GP and DEC-apx-GP require no data exchange to promote federated learning. The third decentralized GP training method (DEC-gapx-GP) addresses the GP training problem for large-scale multi-agent systems by allowing partial data exchnage to provide global information to local datasets. Lastly, we propose a centralized GP training method (gapx-GP) to improve the accuracy of GP hyperparameter estimation for the case of large fleet multi-agent systems, while maintaining scalable computations.

In Section II we overview GP training and discuss the factorized GP training, Section III discusses existing centralized GP training techniques, and Section III-B proposes a new centralized GP training algorithm. In Section IV, we propose methods for decentralized GP training, Section V provides numerical experiments, and Section VI concludes the paper.

II. PRELIMINARIES AND PROBLEM STATEMENT

In this section, we discuss the foundations of algebraic graph theory, overview GP training [6], describe the factorized GP training method [24], and state the problem.

A. FOUNDATIONS

The notation here is standard. The set of all positive real numbers $\mathbb{R}_{>0}$ and the set of all non-negative real numbers $\mathbb{R}_{\geq 0}$. We denote by I_n the identity matrix of $n \times n$ dimension. The vector of n zeros is represented as $\mathbf{0}_n$ and the matrix of $n \times m$ zeros as $\mathbf{0}_{n \times m}$. The superscript in parenthesis $y^{(s)}$ denotes the s-th iteration of an estimation process. The cardinality of the set K is denoted $\mathrm{card}(K)$, the absolute values is denoted $|\cdot|$, the L_2 norm is denoted $\|\cdot\|_2$, and $\|\cdot\|_\infty$ denotes the infinity norm. The notation $\underline{\lambda}(F)$ denotes the minimum eigenvalue of matrix F. The i-th element of a vector x is denoted x_i and x_i denotes the vector x of agent i. A collection of elements of a vector $x \in \mathbb{R}^N$ is denoted $\{x_i\}_{i=1}^N$.

The communication complexity is denoted by $\mathcal{O}(\cdot)$ and describes the total number of bits required to be transmitted over the course of the algorithm up to convergence [19, Chapter 3]. Time and space complexity are also denoted by $\mathcal{O}(\cdot)$ and provide the maximum computations to be performed and space to be occupied at any instant of an algorithm respectively. All complexities are calculated with respect to the total number of observations N, the input dimension D, and the number of agents M. In addition, the communication complexity considers the iterations required for an algorithm to converge s^{end}.

Suppose a network consists of M agents that can perform local computations. The network is described by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = v_1, \dots, v_M$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ the set of edges. Nodes represent agents and edges their communication. The neighbors of the i-th node are denoted $\mathcal{N}_i = \{v_j \in \mathcal{V} \mid (v_i, v_j) \in \mathcal{E}\}$. The adjacency matrix of $\mathcal{G}$ is denoted $A = [a_{ij}] \in \mathbb{R}^{M \times M}$, where $a_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. We consider a decentralized network topology described by a path graph that represents the most parsimonious connected graph [34]. This means that as we increase network connectivity the performance of the proposed algorithms will improve, because we test our methods on the worst case network topology.

Assumption 1. *[35] A graph $\mathcal{G}$ is strongly connected if for every pair of distinct agents (v_i, v_j) there exists a path.*

B. GAUSSIAN PROCESS TRAINING

Let the observations be modeled by,

$$y(x) = f(x) + \epsilon, \tag{1}$$

where $x \in \mathbb{R}^D$ is the input location with D the input space dimension, $f(x) \sim \mathcal{GP}(0, k(x, x'))$ is a zero-mean GP with covariance function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is the i.i.d. measurement noise with variance $\sigma_\epsilon^2 > 0$. We use the separable squared exponential (SSE) covariance function,

$$k(x, x') = \sigma_f^2 \exp\left\{-\frac{1}{2}\sum_{d=1}^{D} \frac{(x_d - x_d')^2}{l_d^2}\right\}, \tag{2}$$

where $\sigma_f^2 > 0$ is the signal variance and $l_d > 0$ the length-scale hyperparameter at the d-th direction of the input space. The goal of GPs is to infer the underlying latent function f given data $\mathcal{D} = \{X, y\}$, where $X = \{x_n\}_{n=1}^N$ the inputs, $y = \{y_n\}_{n=1}^N$ the outputs, and N the number of observations.

1) Training

A GP is trained to find the hyperparameter vector $\boldsymbol{\theta} = (l_1, \dots, l_D, \sigma_f, \sigma_\epsilon)^\intercal \in \Theta \subset \mathbb{R}^{D+2}$ that maximizes the marginal log-likelihood,

$$\mathcal{L} = \log p(y \mid X) = -\frac{1}{2}\left(y^\intercal C_\theta^{-1} y + \log|C_\theta| + N \log 2\pi\right),$$

where $C_\theta = K + \sigma_\epsilon^2 I_N$ is the positive definite covariance matrix with $K = k(X, X) \succeq 0 \in \mathbb{R}^{N \times N}$ the positive semi-definite correlation matrix. The minimization problem employs the negative marginal log-likelihood (NLL) function,

$$\text{(P1)} \quad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad y^\intercal C_\theta^{-1} y + \log|C_\theta| \tag{3a}$$

$$\text{s.to} \quad \boldsymbol{\theta} > \mathbf{0}_{D+2}. \tag{3b}$$

The bound constraints (3b) on the lengthscales l_d ensure that the correlation matrix is positive semi-definite. First-order iterative methods (e.g., conjugate gradient descent) or second-order iterative methods with approximated Hessian (e.g., L-BFGS-B) are widely used to tackle (P1) in (3). Both optimization approaches require the computation of the gradient,

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{2}\text{tr}\left\{\left(C_\theta^{-1} - C_\theta^{-1} y y^\intercal C_\theta^{-1}\right)\frac{\partial C_\theta}{\partial \boldsymbol{\theta}}\right\}. \tag{4}$$

The partial derivative of the covariance matrix $\partial C_\theta / \partial \boldsymbol{\theta}$ depends on the covariance function. For our covariance function selection (2), the partial derivative is provided in Appendix A.

2) Complexity

The time complexity of the training is $\mathcal{O}(N^3)$ for computing the inverse of the covariance matrix of (P1) in (3). Note that only the inverse of the covariance matrix C_θ^{-1} is required to be computed for the training (P1) in (3) and not the logarithm of its determinant $\log|C_\theta|$ [6, Appendix A.4]. The inverse computation of the covariance matrix is performed repeatedly in the optimization (P1) to find the hyperparameters $\hat{\boldsymbol{\theta}}$. After solving (P1) and obtaining the hyperparameter vector $\hat{\boldsymbol{\theta}}$, we store the inverse C^{-1} and N observations, which results in $\mathcal{O}(N^2 + DN)$ space complexity.

C. FACTORIZED GP TRAINING (FACT-GP)

Let each agent i to collect local observations and form the local dataset $\{\mathcal{D}_i = \{X_i, y_i\}\}_{i=1}^M$ corresponding to N_i observations for M agents with $\sum_{i=1}^M N_i = N$ and global dataset $\mathcal{D} = \cup_{i=1}^M \mathcal{D}_i$. All local datasets have the same number of observations, i.e., $N_i = N_j = N/M$ for all $i, j \in \mathcal{V}$ with $i \neq j$. In practice, even if all agents have access to the global dataset $\mathcal{D}$, the GP computational complexity (Section II-B2) is prohibitive if $\mathcal{D}$ is large. Factorized GP training (FACT-GP) [24], [36] assumes a centralized topology, where every entity i communicates to a central node with significant computational and storage resources. The centralized topology arises several problems: i) *security* and *robustness*, as the central node is vulnerable to malicious attacks or even failure; ii) *traffic network congestion*, when all agents communicate their local datasets with the central entity; and iii) *privacy*, because a single entity has access to the global dataset. In addition, for certain cases (e.g., multi-robot systems), distant agents may be subject to communication range limitations, thus the centralized topology may not be feasible.

Assumption 2. *Every agent i can communicate only with agents in its neighborhood $\mathcal{N}_i$ and the communication shall not include any data exchange.*

Assumption 3. *Every agent i can communicate only with agents in its neighborhood $\mathcal{N}_i$ and the communication shall include partial exchange of the local dataset $\mathcal{D}_i$.*

Assumption 2 prohibits the communication of any observation, whereas Assumption 3 allows the communication of a subset of the local dataset $\mathcal{D}_i$. This distinction has been made to propose different methodologies in case that partial communication of the local dataset is permitted.

Assumption 4. *All local sub-models $\mathcal{M}_i$ are statistically independent and local datasets represent distinct areas.*

TABLE 1. Time, Space, and Communication Complexity of GP Training and Factorized GP Training Methods

			FULL-GP [6]	FACT-GP [24]	g-FACT-GP [29]
Local		Time	-	$\mathcal{O}(N^3/M^3)$	$\mathcal{O}(8(N^3/M^3))$
		Space	-	$\mathcal{O}(\xi)$	$\mathcal{O}(2\xi+2(N^2/M^2))$
Global	GD	Space	-	$\mathcal{O}(DM+2M)$	$\mathcal{O}(DM+2M)$
		Comm	-	$\mathcal{O}(s^{\text{end}}(DM+2M))$	$\mathcal{O}(s^{\text{end}}(DM+2M))$
	Final	Time	$\mathcal{O}(N^3)$	-	-
		Space	$\mathcal{O}(N^2+DN)$	$\mathcal{O}(N^2/M)$	$\mathcal{O}(4(N^2/M))$
		Comm	-	$\mathcal{O}(N^2/M)$	$\mathcal{O}(4(N^2/M))$

$\xi = N^2/M^2 + D(N/M)$.

Remark 1. *Assumption 4 is a standard assumption in distributed optimization and leads to problems with a global decomposed objective function that is the sum of all local objective functions [37], [38]. The distinction in areas of local datasets ensures that agents cannot gather information from the same input locations with different observations.*

The factorized GP training relies on Assumption 4. This implies that the global marginal likelihood can be approximated by the product of local likelihoods, which leads to,

$$p(y \mid X) \approx \prod_{i=1}^{M} p_i(y_i \mid X_i), \tag{5}$$

where $p_i(y_i \mid X_i) \sim \mathcal{N}(0, C_{\theta,i})$ is the local marginal likelihood of the i-th node with local covariance matrix $C_{\theta,i} = K_i + \sigma_\epsilon^2 I_{N_i}$ and $K_i = k(X_i, X_i) \in \mathbb{R}^{N_i \times N_i}$. Moreover, the factorized approximation (5) yields a block diagonal approximation of the covariance matrix $C_\theta^{-1} \approx \operatorname{diag}(C_{\theta,1}^{-1}, \dots, C_{\theta,M}^{-1})$. Subsequently, the global marginal log-likelihood is approximated by $\mathcal{L} \approx \sum_{i=1}^{M} \mathcal{L}_i$ (Remark 1) which results in,

$$\log p(y \mid X) \approx \sum_{i=1}^{M} \log p_i(y_i \mid X_i),$$

with local marginal log-likelihood $\mathcal{L}_i = \log p_i(y_i \mid X_i)$,

$$\mathcal{L}_i = -\frac{1}{2}\left(y_i^\intercal C_{\theta,i}^{-1} y_i + \log|C_{\theta,i}| + N_i \log 2\pi\right). \tag{6}$$

The gradient of the global log-likelihood in FACT-GP is computed by $\nabla_{\boldsymbol{\theta}} \mathcal{L} = \sum_{i=1}^{M} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i$ [5], [27]. The optimization uses the local negative marginal log-likelihood (LNLL),

$$\text{(P2)} \quad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad \sum_{i=1}^{M} y_i^\intercal C_{\theta,i}^{-1} y_i + \log|C_{\theta,i}| \tag{7a}$$

$$\text{s.to} \quad \boldsymbol{\theta}_i > \mathbf{0}_{D+2}, \qquad \forall i \in \mathcal{V}, \tag{7b}$$

where $\boldsymbol{\theta}_i = \{l_{1,i}, \dots, l_{D,i}, \sigma_{f,i} \sigma_{\epsilon,i}\}$ the local hyperparameters of agent i. Similarly to (3), constraint (7b) imposes positivity on the agreed hyperparameters for all agents $i \in \mathcal{V}$.

The computation of (6) for the FACT-GP (7) yields $\mathcal{O}(N_i^3) = \mathcal{O}(N^3/M^3)$ time complexity for each local entity to invert the local covariance matrix $C_{\theta,i}^{-1}$. Additionally, for the storage of the local inverted covariance matrix and the local observations $\mathcal{O}(N_i^2 + DN_i) = \mathcal{O}(N^2/M^2 + D(N/M))$ space is needed. The factorized training requires communication from every node i to the central node. Provided that the central node implements gradient descent, every node communicates the local gradient of LNLL $\nabla_{\boldsymbol{\theta}} \mathcal{L}_i$ at every iteration s. That is $\mathcal{O}(s^{\text{end}}(D+2)M) = \mathcal{O}(s^{\text{end}}(DM+2M))$ total communications from all agents to the central node, where s^{end} is the total number of iterations to reach convergence. Additionally, the central node needs to store at each iteration: i) the hyperparameter vector on the previous iteration $\{\boldsymbol{\theta}_i^{(s)}\}_{i=1}^M$ from all M nodes; and ii) their gradient of LNLL $\{\nabla_{\boldsymbol{\theta}} \mathcal{L}_i\}_{i=1}^M$, which results in $\mathcal{O}((D+2)M + (D+2)) = \mathcal{O}(DM + 2M)$ space complexity. Finally, after the optimization algorithm converges, each node communicates the local inverted covariance matrix $C_{\theta,i}^{-1}$ that yields $\mathcal{O}(MN_i^2) = \mathcal{O}(N^2/M)$ communications to the central node. All local inverted covariance matrices need to be stored in the central node leading to $\mathcal{O}(N^2/M)$ space complexity. A computational complexity comparison between FULL-GP and FACT-GP is provided in Table 1. Since $N_i = N/M < N$, FACT-GP requires less time and space than FULL-GP.

D. PROBLEM DEFINITION

Problem 1. *Under Assumption 2 and 3, solve the optimization problem (P2) in* (7) *to estimate the GP hyperparmeters* $\hat{\boldsymbol{\theta}}$ *for a connected decentralized network topology (Assumption 1) with independent local GP models (Assumption 4).*

Problem 1 is twofold with partial data exchange and no data exchange between agents. In particular, Assumption 2 allows no data exchange to satisfy federated learning, whereas Assumption 3 relaxes the problem by allowing partial data exchange. Both versions of Problem 1 consider a connected decentralized network (Assumption 1) and take the independence approximation assumption between local datasets (Assumption 4). Recent advancements in distributed GP training [5], [27] have addressed the centralized version of Problem 1. In Section III we review [5], [27] and propose an extension for centralized networks. However, the main focus of this paper is on decentralized networks without requiring a central coordinator with massive computational and storage capabilities. In Section IV, we propose the first decentralized methods to perform GP training for both versions of Problem 1.

III. CENTRALIZED GP TRAINING

In this section, we discuss two existing centralized GP training methods [5], [27] that reduce the computational complexity of FACT-GP (7) based on the alternating direction method of multipliers (ADMM) [4]. Next, we propose a centralized GP training method that is efficient for large fleet multi-agent systems. In addition, we derive all computational and communication complexities to compare existing methods with the proposed centralized method.

The following Assumptions are required for first-order approximations and convergence properties.

Assumption 5. *A function $f:\mathbb{R}^{\mathrm{N}}\to\mathbb{R}$ is Lipschitz continuous with positive parameter $L>0$ if it satisfies,*

$$\|\nabla f(x)-\nabla f(y)\|_2\leq L\|x-y\|_2,\quad \forall x,y. \tag{8}$$

Assumption 6. *A function $f:\mathbb{R}^{\mathrm{N}}\to\mathbb{R}$ is strongly convex with positive parameter $m>0$ if it satisfies,*

$$(\nabla f(x)-\nabla f(y))^\intercal(x-y)\geq m\|x-y\|_2^2,\quad \forall x,y. \tag{9}$$

Remark 2. *Assumption 6 requires the local log-likelihood function $\mathcal{L}_i$ to be strongly convex. Usually $\mathcal{L}_i$ is nonconvex with respect to the hyperparameters $\boldsymbol{\theta}_i$ [6], [39], [40]. This is a well known issue of GP training with MLE. A common trick to address the nonconvexity problem is to use multiple starting points to the optimization [6], [41], [42]. However, as we increase the dataset size, the local log-likelihoods tend to be unimodal distributions [39], and thus Assumption 6 holds.*

A. EXISTING CENTRALIZED GP TRAINING METHODS

To address the centralized factorized GP training problem (7) an exact consensus ADMM and an inexact proximal consensus ADMM have been used in [5], [27]. Using the logarithmic transformation, (7) can be expressed as,

$$\text{(P3)}\quad \hat{\boldsymbol{\theta}}=\arg\min_{\boldsymbol{\theta}}\ \sum_{i=1}^{M}y_i^\intercal C_{\theta,i}^{-1}y_i+\log|C_{\theta,i}| \tag{10a}$$

$$\text{s.to}\quad \boldsymbol{\theta}_i=z,\quad \forall i\in\mathcal{V}, \tag{10b}$$

where $\boldsymbol{\theta}_i=\{l_{1,i},\dots,l_{D,i},\sigma_{f,i}\sigma_{\epsilon,i}\}$ is the local vector of hyperparameters of agent i, and $z\in\mathbb{R}^{D+2}$ is an auxiliary variable. In other words, constraint (10b) implies that every agent i is allowed to have its own opinion for the hyperparameters $\boldsymbol{\theta}_i$, yet at the end of the optimization all agents must agree on the global vector value z. Recognize that (10) has the same formulation with the exact consensus ADMM problem [4]. Thus, after formulating the augmented Lagrangian, the consensus GP training (c-GP) iterative scheme [27] yields,

$$z^{(s+1)}=\frac{1}{M}\sum_{i=1}^{M}\left(\boldsymbol{\theta}_i^{(s)}+\frac{1}{\rho}\boldsymbol{\psi}_i^{(s)}\right), \tag{11a}$$

$$\boldsymbol{\theta}_i^{(s+1)}=\arg\min_{\boldsymbol{\theta}_i}\left\{\mathcal{L}_i(\boldsymbol{\theta}_i)+\left(\boldsymbol{\psi}_i^{(s)}\right)^\intercal\left(\boldsymbol{\theta}_i-z^{(s+1)}\right)+\frac{\rho}{2}\left\|\boldsymbol{\theta}_i-z^{(s+1)}\right\|_2^2\right\}, \tag{11b}$$

$$\boldsymbol{\psi}_i^{(s+1)}=\boldsymbol{\psi}_i^{(s)}+\rho\left(\boldsymbol{\theta}_i^{(s+1)}-z^{(s+1)}\right), \tag{11c}$$

where $\boldsymbol{\psi}_i\in\mathbb{R}^{D+2}$ is the vector of dual variables of node i, $s\in\mathbb{Z}_{\geq0}$ is the iteration number, and $\rho>0$ is the penalty constant term of the augmented Lagrangian.

Let $s_{\text{nest}}^{\text{end}}$ be the number of iterations required from the nested optimization (11b) to converge. The computational complexity of c-GP is cubic in the number of local observations $\mathcal{O}(s_{\text{nest}}^{\text{end}}N_i^3)=\mathcal{O}(s_{\text{nest}}^{\text{end}}(N^3/M^3))$. The nested optimization problem (11b) requires the evaluation of the local log-likelihood $\mathcal{L}_i(\boldsymbol{\theta}_i)$ at every internal iteration s_{nest} which entails cubic computations to invert the local covariance $C_{\theta,i}^{-1}$ (6). The communication complexity to transmit all local hyperparameter vectors yields $\mathcal{O}(s^{\text{end}}M(D+2))$. After convergence, every agent i transmits the local inverted covariance $C_{\theta,i}^{-1}$ requiring $\mathcal{O}(MN_i^2)=\mathcal{O}(N^2/M)$ communications. Every agent i occupies $\mathcal{O}(N_i^2+3(D+2)+D(N/M)))=\mathcal{O}(N^2/M^2+DN/M)$ memory to store $C_{\theta,i}^{-1}$, $\boldsymbol{\theta}_i^{(s)}$, $z^{(s)}$, $\boldsymbol{\psi}_i^{(s)}$, and $\mathcal{D}_i$.

The major disadvantage of c-GP is the time complexity of the nested optimization problem (11b). To address this issue, the authors in [5] employed the proximal consensus ADMM [28] and derived an analytical solution for the case of centralized factorized GP training to form the analytical proximal GP training (apx-GP). Note that apx-GP employs a first-order approximation (linearization) on the local log-likelihood $\mathcal{L}_i$ around $z^{(s+1)}$,

$$\mathcal{L}_i(\boldsymbol{\theta}_i)\approx\nabla_{\boldsymbol{\theta}}^\intercal\mathcal{L}_i\left(z^{(s+1)}\right)\left(\boldsymbol{\theta}_i-z^{(s+1)}\right)+\frac{L_i}{2}\left\|\boldsymbol{\theta}_i-z^{(s+1)}\right\|_2^2, \tag{12}$$

where $L_i>0$ is a positive Lipschitz constant that satisfies Assumption 5 of the local log-likelihood function $\mathcal{L}_i$ for all $i\in\mathcal{V}$. The apx-GP iteration steps [5] are given by,

$$z^{(s+1)}=\frac{1}{M}\sum_{i=1}^{M}\left(\boldsymbol{\theta}_i^{(s)}+\frac{1}{\rho}\boldsymbol{\psi}_i^{(s)}\right), \tag{13a}$$

$$\boldsymbol{\theta}_i^{(s+1)}=z^{(s+1)}-\frac{1}{\rho+L_i}\left(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i\left(z^{(s+1)}\right)+\boldsymbol{\psi}_i^{(s)}\right) \tag{13b}$$

$$\boldsymbol{\psi}_i^{(s+1)}=\boldsymbol{\psi}_i^{(s)}+\rho\left(\boldsymbol{\theta}_i^{(s+1)}-z^{(s+1)}\right), \tag{13c}$$

where the gradient of the local log-likelihood $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$ has similar structure to the the gradient of the log-likelihood (4). The only difference on the workflow of apx-GP and c-GP is that the second step of $\boldsymbol{\theta}_i^{(s+1)}$ is computed analytically (13b), while the former incorporates a nested optimization problem (11b) at every ADMM-iteration.

The space and communication complexity of apx-GP is identical to c-GP. The time complexity of apx-GP entails

TABLE 2. Time, Space, and Communication Complexity of Centralized GP Training Methods

			c-GP [27]	apx-GP [5]	gapx-GP (proposed)
Local		Time	$\mathcal{O}(s_{\text{nest}}^{\text{end}}(N^3/M^3))$	$\mathcal{O}(N^3/M^3)$	$\mathcal{O}(8(N^3/M^3))$
		Space	$\mathcal{O}(\xi)$	$\mathcal{O}(\xi)$	$\mathcal{O}(2\xi + 2(N^2/M^2))$
Global	ADMM	Comm	$\mathcal{O}(s_{\text{c-GP}}^{\text{end}}M(D+2))$	$\mathcal{O}(s_{\text{apx-GP}}^{\text{end}}M(D+2))$	$\mathcal{O}(s_{\text{gapx-GP}}^{\text{end}}M(D+2))$
	Final	Comm	$\mathcal{O}(N^2/M)$	$\mathcal{O}(N^2/M)$	$\mathcal{O}(4(N^2/M))$

$\xi = N^2/M^2 + D(N/M)$.

$\mathcal{O}(N_i^3) = \mathcal{O}(N^3/M^3)$ computations, significantly reduced from $\mathcal{O}(s_{\text{nest}}^{\text{end}}N^3/M^3)$ of c-GP. In other words, there is no nested optimization problem in apx-GP (13). Thus, apx-GP requires just one inversion of the local covariance matrix $C_{\theta,i}^{-1}$ per ADMM-iteration instead of $s_{\text{nest}}^{\text{end}}$ inversions per ADMM-iteration of c-GP. Both c-GP and apx-GP inherit fast convergence properties of the exact consensus ADMM [4] and the inexact proximal consensus ADMM [28].

A disadvantage of both centralized methods (c-GP (11) and apx-GP (13)) is that they are based on factorized GP training and thus they inherit poor approximation capabilities when the number of agent increases. In other words, for a bounded space of interest, Assumption 4 is violated as we increase the number of sub-models $\mathcal{M}_i$. In what follows, we seek to improve the approximation error for large scale multi-agent systems by inheriting global information to all local $\mathcal{D}_i$.

B. PROPOSED CENTRALIZED GP TRAINING

The first method we propose is a centralized factorized GP training technique that extends apx-GP with a local augmented datatset $\mathcal{D}_{+i}$ for all $i \in \mathcal{V}$. The goal is to limit the approximation error of apx-GP for large fleet sizes inherited by Assumption 4 at the cost of allowing partial data exchange (Assumption 3). Data exchange leads to larger datasets that entail higher computations. In other words, we aim to improve GP hyperparameter estimation accuracy for centralized large fleet networks with higher yet reasonable computations. Our methodology is termed as generalized apx-GP (gapx-GP).

The main idea of gapx-GP is to equip every agent with a new dataset that has global information on the underlying latent function. Every agent i selects randomly without replacement N_i/M data from its local dataset $\mathcal{D}_i$ to form the *local sample dataset* $\mathcal{D}_{-i} \in \mathbb{R}^{N_i/M} \subset \mathcal{D}_i$. Then, the local sample datasets are communicated to every other agent (Assumption 3) to compose the *communication dataset* $\mathcal{D}_{\text{c}} = \{\mathcal{D}_{\text{-i}}\}_{i=1}^{M} = \{X_{\text{c}}, y_{\text{c}}\}$. Next, every agent i fuses the communication dataset $\mathcal{D}_{\text{c}} \in \mathbb{R}^{N_i}$ with its local dataset $\mathcal{D}_i$ to form the *local augmented dataset* $\mathcal{D}_{+i} = \mathcal{D}_i \cup \mathcal{D}_{\text{c}} \in \mathbb{R}^{2N_i}$. The local augmented dataset $\mathcal{D}_{+i}$ is a new dataset for every agent i that includes the local dataset $\mathcal{D}_i$ and the communication dataset $\mathcal{D}_{\text{c}}$, providing a global representation. Note that in [29], the communication dataset $\mathcal{D}_{\text{c}}$ is randomly selected from the full dataset $\mathcal{D}$, while we consider a slight variation for decentralized networks. In other words, the communication dataset $\mathcal{D}_{\text{c}}$ is selected by the local datasets $\mathcal{D}_i$ and then fused through

Algorithm 1 gapx-GP

```
Input: $\mathcal{D}_i(X_i, y_i)$, $k(\cdot,\cdot)$, $\rho$, $L_i$, $\mathcal{N}_i$, $\mathcal{V}$, TOL$_{\text{ADMM}}$
Output: $\hat{\boldsymbol{\theta}}$, $C_{\theta}^{-1}$, $\mathcal{D}_{+i}$
 1: for each $i \in \mathcal{V}$ do                                  ▷ Local Sample Dataset
 2:     $\mathcal{D}_{\text{c},i} \leftarrow$ Sample($\mathcal{D}_i$)
 3:     communicate $\mathcal{D}_{\text{c},i}$ to central node
 4: end for
 5: scatter $\mathcal{D}_{\text{c}} = \{\mathcal{D}_{\text{c},i}\}_{i=1}^{M}$ from central node to every agent
 6: for each $i \in \mathcal{V}$ do                                  ▷ Local Augmented Dataset
 7:     $\mathcal{D}_{+i} \leftarrow \mathcal{D}_i \cup \mathcal{D}_{\text{c}}$
 8: end for
 9: repeat                                                          ▷ ADMM Optimization
10:     communicate $\boldsymbol{\theta}_i^{(s)}$ to central node
11:     $z^{(s+1)} \leftarrow$ prim-2($\boldsymbol{\theta}_i^{(s)}, \boldsymbol{\psi}_i^{(s)}$, card($\mathcal{V}$)) (13a)
12:     scatter $z^{(s+1)}$ from central node to every agent
13:     for each $i \in \mathcal{V}$ do
14:         $\boldsymbol{\theta}_i^{(s+1)} \leftarrow$ prim-1($\boldsymbol{\theta}_i^{(s)}, z^{(s+1)}, \boldsymbol{\psi}_i^{(s)}, \rho, L_i, \mathcal{D}_{+i}$) (13b)
15:         $\boldsymbol{\psi}_i^{(s+1)} \leftarrow$ dual($\boldsymbol{\theta}_i^{(s+1)}, z^{(s+1)}, \boldsymbol{\psi}_i^{(s)}, \rho$) (13c)
16:     end for
17: until $\|\boldsymbol{\theta}_i^{(s+1)} - z^{(s+1)}\|_2 <$ TOL$_{\text{ADMM}}$, for all $i \in \mathcal{V}$
18: for each $i \in \mathcal{V}$ do              ▷ Local Augmented Covariance Inversion
19:     $\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}_i^{\text{end}}$
20:     $C_{\theta,+i}^{-1} \leftarrow$ invert($k, X_{+i}, \hat{\boldsymbol{\theta}}$)
21:     communicate $C_{\theta,+i}^{-1}$ to central node
22: end for
23: $C_{\theta}^{-1} \leftarrow$ diag($C_{\theta,+1}^{-1}, C_{\theta,+2}^{-1}, \ldots, C_{\theta,+M}^{-1}$)     ▷ Block Diagonal
24: Return $\hat{\boldsymbol{\theta}}$, $C_{\theta}^{-1}$, $\mathcal{D}_{+i}$
```

information exchange. Next, we implement the apx-GP (13), but now every agent is equipped with the local augmented dataset $\mathcal{D}_{+i}$ (Algorithm 1).

The local time complexity of gapx-GP yields $\mathcal{O}((2N_i)^3) = \mathcal{O}(8(N^3/M^3))$ computations to invert the local augmented covariance matrix $C_{\theta,+i} = K_{+i} + \sigma_\epsilon^2 I_{2N_i} \in \mathbb{R}^{2N_i \times 2N_i}$. The total communication overhead is the same with c-GP and apx-GP. After convergence, each agent i communicates the local augmented covariance matrix $C_{\theta,+i}^{-1}$ that entails $\mathcal{O}(M(2N_i)^2) = \mathcal{O}(4(N^2/M))$ communications. The space complexity of every agent i yields $\mathcal{O}((2N_i)^2 + 3(D+2) + D(2N_i)) = \mathcal{O}(4(N^2/M^2) + 2D(N/M))$ to store the local augmented covariance matrix, the optimization variables at the previous iteration, and the local augmented dataset.

In Table 2, we list the time, space, and communication complexity for all centralized factorized GP training methods based on ADMM. The proposed method is more demanding in space than c-GP. In terms of time complexity, gapx-GP is more affordable than c-GP, because the nested optimization

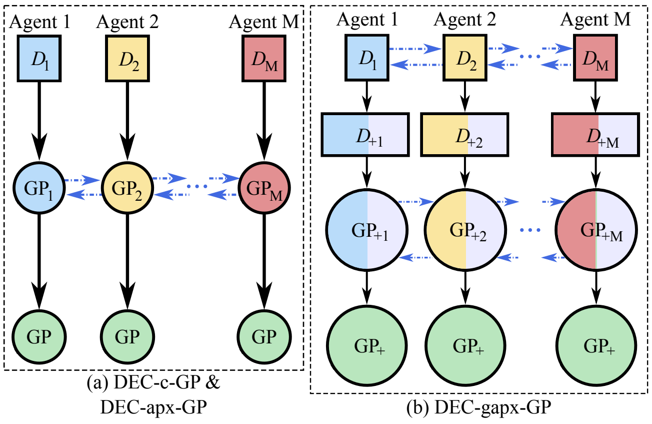

FIGURE 2. The structure of the proposed decentralized GP training methods. Blue dotted lines correspond to communication (strongly connected). a) Every agent *i* has access to the local dataset $\mathcal{D}_i$. b) Every agent *i* has access to $\mathcal{D}_i$. Next, they communicate to form the local augmented dataset $\mathcal{D}_{+i}$ which comprises of $\mathcal{D}_i$ (local color) and the global communication dataset $\mathcal{D}_c$ (gray color).

of the latter (11b) takes on average more than eight iterations to converge, i.e., $s_{\text{nest}}^{\text{end}} > 8$, but more demanding than apx-GP. The proposed method supports Assumption 4, and thus we expect to produce more accurate hyperparameters.

Proposition 1. *[28, Theorem 2.10] Consider a strongly connected decentralized network (Assumption 1) where the agents are allowed to communicate partially their datasets (Assumption 3). Let Assumption 4 hold for the local sub-models $\mathcal{M}_i$ and Assumption 6 hold for all local log-likelihoods $\mathcal{L}_i$ (6), then the gapx-GP converges $\lim_{s\to\infty}\|\boldsymbol{\theta}_i^{(s)} - z^{(s)}\|_D = 0$ to a stationary solution $(\boldsymbol{\theta}_i^\star, z^\star, \psi_i^\star)$ for all agents $i \in \mathcal{V}$.* □

Proposition 1 implies that the convergence properties of the optimization scheme (13) hold for gapx-GP to address (P3) in (10). In other words, the proposed centralized gapx-GP equipped with the local augmented datasets $\mathcal{D}_{+i}$ converges to the optimal hyperparameters $\boldsymbol{\theta}_i^\star$ for all agents $i \in \mathcal{V}$.

IV. PROPOSED DECENTRALIZED GP TRAINING

In this section, we introduce three methods to address Problem 1 based on the *edge formulation* of ADMM [43] that yields parallel updates and decentralizes the factorized GP training. Algorithmic implementation details are discussed for all proposed methods. In addition, we provide a time, space, and communication complexity analysis.

The edge formulation is a variation (10) for decentralized networks. Let Assumption 1 hold, then (10) yields,

$$\text{(P4)}\quad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{M} y_i^\intercal C_{\theta,i}^{-1} y_i + \log|C_{\theta,i}| \tag{14a}$$

$$\text{s.to}\quad \boldsymbol{\theta}_i = \boldsymbol{\tau}_{ij},\quad \forall i \in \mathcal{V}, j \in \mathcal{N}_i, \tag{14b}$$

$$\boldsymbol{\theta}_j = \boldsymbol{\tau}_{ij},\quad \forall i \in \mathcal{V}, j \in \mathcal{N}_i, \tag{14c}$$

where $\boldsymbol{\tau}_{ij}$ are auxiliary variables. Constraints (14b) and (14c) imply that every agent i is allowed to have its own opinion

Algorithm 2 DEC-c-GP

Input: $\mathcal{D}_i(X_i, y_i)$, $k(\cdot,\cdot)$, ρ, $\mathcal{N}_i$, α, $s_{\text{DEC-c-GP}}^{\text{end}}$
Output: $\hat{\boldsymbol{\theta}}$, $C_{\theta,i}^{-1}$

```
1:  initialize $p_i^{(0)} = \mathbf{0}$
2:  for $s = 1$ to $s_{\text{DEC-c-GP}}^{\text{end}}$ do                ▷ ADMM Optimization
3:      for each $i \in \mathcal{V}$ do
4:          communicate $\boldsymbol{\theta}_i^{(s)}$ to neighbors $\mathcal{N}_i$
5:          $p_i^{(s+1)} \leftarrow \texttt{duals}(p_i^{(s)}, \boldsymbol{\theta}_i^{(s)}, \{\boldsymbol{\theta}_j^{(s)}\}_{j\in\mathcal{N}_i}, \rho)$ (15a)
6:          $\boldsymbol{\theta}_i^{(s+1)} \leftarrow \texttt{prim}(p_i^{(s+1)}, \boldsymbol{\theta}_i^{(s)}, \{\boldsymbol{\theta}_j^{(s)}\}_{j\in\mathcal{N}_i}, \rho, \alpha, \mathcal{D}_i)$ (15b)
7:      end for
8:  end for
9:  for each $i \in \mathcal{V}$ do                ▷ Local Covariance Inversion
10:     $\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}_i^{\text{end}}$
11:     $C_{\theta,i}^{-1} \leftarrow \texttt{invert}(k, X_i, \hat{\boldsymbol{\theta}})$
12: end for
13: Return $\hat{\boldsymbol{\theta}}$, $C_{\theta,i}^{-1}$
```

Algorithm 3 DEC-apx-GP

Input: $\mathcal{D}_i(X_i, y_i)$, $k(\cdot,\cdot)$, ρ, $\mathcal{N}_i$, κ_i, $s_{\text{DEC-apx-GP}}^{\text{end}}$
Output: $\hat{\boldsymbol{\theta}}$, $C_{\theta,i}^{-1}$

```
1: Identical to Algorithm 2 with (15b) replaced by (19b)
```

for the hyperparameters $\boldsymbol{\theta}_i$, yet at the end of the optimization all agents in the neighborhood $\mathcal{N}_i$ must agree on the neighborhood values $\boldsymbol{\tau}_{ij}$. The edge formulation requires each node i to store and update variables for all of its neighbors $\mathcal{N}_i$. Conversely, one can employ the *node formulation* that relaxes the storage capacity, as each agent i is required to store and update variables of itself [44]. In addition, the group ADMM [45] offers a decentralized optimization method, yet for a specific graph topology. Thus, we find the edge formulation more suitable for decentralized GP training.

A. PROPOSED DEC-C-GP

The first proposed method is based on the decentralized consensus ADMM [46] to perform GP training (DEC-c-GP). After rendering the augmented Lagrangian for (P4) in (14) we obtain the decentralized consensus ADMM iterative scheme,

$$p_i^{(s+1)} = p_i^{(s)} + \rho \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\theta}_i^{(s)} - \boldsymbol{\theta}_j^{(s)}\right), \tag{15a}$$

$$\boldsymbol{\theta}_i^{(s+1)} = \arg\min_{\boldsymbol{\theta}_i} \Bigg\{ \mathcal{L}_i(\boldsymbol{\theta}_i) + \boldsymbol{\theta}_i^\intercal p_i^{(s+1)} + \rho \sum_{j\in\mathcal{N}_i} \left\| \boldsymbol{\theta}_i - \frac{\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)}}{2} \right\|_2^2 \Bigg\}, \tag{15b}$$

where $\rho > 0$ is the penalty term of the augmented Lagrangian and $p_i^{(s)} = \sum_{j\in\mathcal{N}_i}(u_{ij}^{(s)} + v_{ij}^{(s)})$ is the sum of the dual variables $u_{ij}^{(s)}$ and $v_{ij}^{(s)}$ corresponding to constraints (14b) and (14c). Note that (15a) imposes initial values $p_i^{(0)} = \mathbf{0}$.

The workflow is as follows. Every agent i communicates to its neighbors $j \in \mathcal{N}_i$ the current estimate of the hyperparameters $\boldsymbol{\theta}_i^{(s)}$. After each agent gathers all $\boldsymbol{\theta}_j^{(s)}$ vectors from its neighborhood, then the sum of the dual variables vector

TABLE 3. Time, Space, and Communication Complexity of Decentralized GP Training Methods

		DEC-c-GP	DEC-apx-GP	DEC-gapx-GP
Local	Time	$\mathcal{O}(s_{\text{nest}}^{\text{end}}(N^3/M^3))$	$\mathcal{O}(N^3/M^3)$	$\mathcal{O}(8(N^3/M^3))$
	Space	$\mathcal{O}(\xi)$	$\mathcal{O}(\xi)$	$\mathcal{O}(2\xi + 2(N^2/M^2))$
	Comm	$\mathcal{O}(s_{\text{DEC-c-GP}}^{\text{end}}(D+2))$	$\mathcal{O}(s_{\text{DEC-apx-GP}}^{\text{end}}(D+2))$	$\mathcal{O}(s_{\text{DEC-gapx-GP}}^{\text{end}}(D+2))$

$\xi = N^2/M^2 + D(N/M)$.

is updated (15a) to obtain $p_i^{(s+1)}$. Next, every agent i solves a nested optimization problem (15b) to compute $\boldsymbol{\theta}_i^{(s+1)}$. The method iterates until it reaches a predefined maximum iteration number $s_{\text{DEC-c-GP}}^{\text{end}}$. The main routine of DEC-c-GP is provided in Algorithm 2. The proposed method is decentralized, requiring exclusively neighbor-wise communication as shown in Fig. 2-(a). Note that the inter-agent communications do not involve any data exchange which satisfies Assumption 2. Provided that the graph topology is connected (Assumption 1), then DEC-c-GP (15) addresses Problem 1.

Proposition 2. *[46, Proposition 2] Consider a strongly connected decentralized network (Assumption 1) where the agents are not allowed to communicate their datasets (Assumption 2). Let Assumption 4 hold for the local sub-models $\mathcal{M}_i$ and Assumption 6 hold for all local log-likelihoods $\mathcal{L}_i$ (6), then the DEC-c-GP* (15) *converges to a stationary solution $\lim_{s\to\infty} \boldsymbol{\theta}_i^{(s)} = \boldsymbol{\theta}^\star$ for all agents $i \in \mathcal{V}$.* □

Proposition 2 implies that the convergence properties of the optimization scheme (15) hold for DEC-c-GP to address (14). In other words, the proposed decentralized DEC-c-GP converges to the optimal hyperparameters $\boldsymbol{\theta}_i^\star$ for all $i \in \mathcal{V}$.

Remark 3. *A disadvantage of the proposed DEC-c-GP is the cubic computations on the number of local observations for every iteration of the nested optimization. That is because at every ADMM iteration we need to solve the nested optimization problem* (15b) *which entails the computation of $\mathcal{L}_i$* (6) *that involves the inversion of the local covariance matrix $C_{\theta,i}^{-1}$.*

B. PROPOSED DEC-APX-GP

To address the computational scalability of DEC-c-GP (Remark 3) we employ the decentralized inexact proximal consensus ADMM [47] and derive an analytical solution to perform GP training (DEC-apx-GP). A proximal step is taken based on a first-order approximation on the local log-likelihood $\mathcal{L}_i$ around $\boldsymbol{\theta}^{(s)}$,

$$\mathcal{L}_i(\boldsymbol{\theta}_i) \approx \nabla_{\boldsymbol{\theta}}^{\intercal}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right)\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right) + \frac{\kappa_i}{2}\left\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right\|_2^2, \tag{16}$$

where $\kappa_i > 0$ is a penalty parameter of the proximal term for all $i \in \mathcal{V}$ and $\nabla_{\boldsymbol{\theta}}^{\intercal}\mathcal{L}_i$ can be computed as in (20). After rendering the augmented Lagrangian in (14) we obtain the decentralized inexact proximal consensus ADMM iterative scheme,

$$p_i^{(s+1)} = p_i^{(s)} + \rho \sum_{j\in\mathcal{N}_i}\left(\boldsymbol{\theta}_i^{(s)} - \boldsymbol{\theta}_j^{(s)}\right), \tag{17a}$$

$$\begin{aligned}\boldsymbol{\theta}_i^{(s+1)} = \arg\min_{\boldsymbol{\theta}_i} \Bigg\{ &\nabla_{\boldsymbol{\theta}}^{\intercal}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right)\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right) + \\ &\frac{\kappa_i}{2}\left\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right\|_2^2 \boldsymbol{\theta}_i^{\intercal} p_i^{(s+1)} + \\ &\rho\sum_{j\in\mathcal{N}_i}\left\|\boldsymbol{\theta}_i - \frac{\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)}}{2}\right\|_2^2\Bigg\}. \end{aligned}\tag{17b}$$

The linearization (16) allows the evaluation of the local log-likelihood function $\mathcal{L}_i$ (6) at a fixed point $\boldsymbol{\theta}_i^{(s)}$ and not at the optimizing variable $\boldsymbol{\theta}_i$. To this end, the nested optimization of (17b) entails significantly less computations than (15b), because we need to compute $\nabla_{\boldsymbol{\theta}}^{\intercal}\mathcal{L}_i(\boldsymbol{\theta}_i^{(s)})$ just one time in (17b) and not at every iteration of the nested optimization problem (15b) (Remark 3). In the following Theorem, we extend [47] by deriving a closed-form solution for the nested optimization (17b) that reduces significantly the computations.

Theorem 1. *Consider a strongly connected decentralized network (Assumption 1) where the agents are not allowed to communicate their datasets (Assumption 2). Let Assumption 4 hold for the local sub-models $\mathcal{M}_i$, Assumption 6 hold for all local log-likelihoods $\mathcal{L}_i$* (6)*, and allow the penalty term of the first-order approximation κ_i to be sufficiently large,*

$$\kappa_i > \frac{L_i^2}{m_i^2} - \rho\underline{\lambda}(D+A) > 0, \quad \forall i \in \mathcal{V}. \tag{18}$$

Then, the nested optimization for the hyperparameter update (17b) *admits a closed-form solution, resulting in the iterative optimization scheme of DEC-apx-GP,*

$$p_i^{(s+1)} = p_i^{(s)} + \rho \sum_{j\in\mathcal{N}_i}\left(\boldsymbol{\theta}_i^{(s)} - \boldsymbol{\theta}_j^{(s)}\right), \tag{19a}$$

$$\begin{aligned}\boldsymbol{\theta}_i^{(s+1)} = \frac{1}{\kappa_i + 2\text{card}(\mathcal{N}_i)\rho}\Bigg(&\rho\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_j^{(s)} - \nabla_{\boldsymbol{\theta}}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) + \\ &\left(\kappa_i + \text{card}(\mathcal{N}_i)\rho\right)\boldsymbol{\theta}_i^{(s)} - p_i^{(s+1)}\Bigg),\end{aligned}\tag{19b}$$

that converges to a stationary $(\boldsymbol{\theta}_i^\star, p^\star)$ for all agents $i \in \mathcal{V}$.

Proof. The proof is provided in Appendix C. □

Remark 4. *The condition to select the penalty parameter κ_i* (18) *depends on the graph topology as the minimum eigenvalue of the degree and adjacency matrix is required $\underline{\lambda}(D+A)$.*

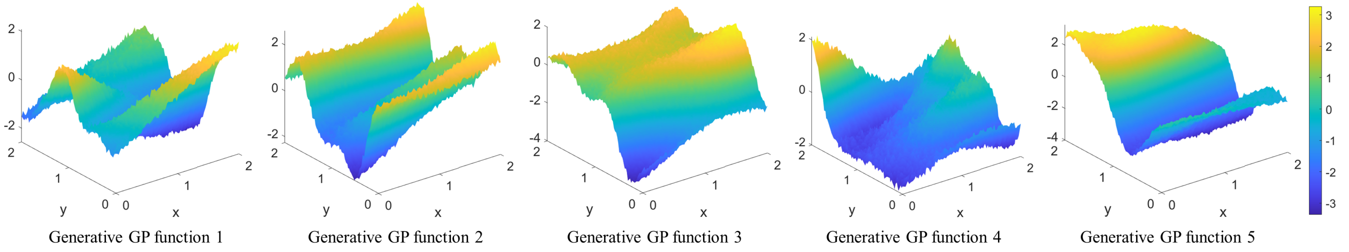

FIGURE 3. Five random function generations of the synthetic GP with known hyperparameter values $\theta = (1.2, 0.3, 1.3, 0.1)^\intercal$ for $N = 8,100$ data.

Algorithm 4 DEC-gapx-GP

```
Input: $\mathcal{D}_i(X_i, y_i)$, $k(\cdot,\cdot)$, $\rho$, $\mathcal{N}_i$, $\kappa_i$, $s^{\text{end}}_{\text{DEC-gapx-GP}}$
Output: $\hat{\boldsymbol{\theta}}$, $C^{-1}_{\theta,+i}$, $\mathcal{D}_{+i}$
1: for each $i \in \mathcal{V}$ do
2:     $\mathcal{D}_{\text{c},i} \leftarrow$ Sample($\mathcal{D}_i$)
3:     $\mathcal{D}_{\text{c}} \leftarrow$ flooding($\mathcal{D}_{\text{c},i}$)
4:     $\mathcal{D}_{+i} = \mathcal{D}_i \cup \mathcal{D}_{\text{c}}$
5: end for
6: $C^{-1}_{\theta,+i} \leftarrow$ DEC-apx-GP($\mathcal{D}_{+i}, k, \rho, \mathcal{N}_i, \kappa_i, s^{\text{end}}_{\text{DEC-gapx-GP}}$)
7: Return $\hat{\boldsymbol{\theta}}$, $C^{-1}_{\theta,+i}$, $\mathcal{D}_{+i}$
```

Thus, the stronger the network connectivity, the faster the convergence of the proposed DEC-apx-GP (19).

The workflow of DEC-apx-GP is similar to DEC-c-GP, yet the hyperparameter update step (19b) is performed analytically without requiring a nested optimization update as in (15b) or (17b). Implementation details are given in Algorithm 3 and the structure is illustrated in Fig. 2-(a). The gradient of the local log-likelihood $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$ is provided in Appendix B-(20).

Remark 5. *A disadvantage of both decentralized methods DEC-c-GP and DEC-apx-GP is the poor approximation capabilities when the number of agents increases. In particular, Assumption 4 is violated as we increase the number of sub-models $\mathcal{M}_i$, which leads to inaccurate GP hyperparameter estimation for large fleet multi-agent systems.*

C. PROPOSED DEC-GAPX-GP

We propose to extend the computationally efficient DEC-apx-GP method (Section IV-B) with a local augmented dataset $\mathcal{D}_{+i}$ for all $i \in \mathcal{V}$ to address the poor approximation capabilities of (19) when the network has large number of agents (Remark 5). The idea is similar to the centralized gapx-GP method (Section III-B). In order to reduce the approximation error, we relax Assumption 2 by allowing partial exchange of local subsets of data (Assumption 3). We term the proposed method as generalized DEC-apx-GP (DEC-gapx-GP).

Since the network has a decentralized topology, flooding [48] is employed to broadcast the local sample datasets $\mathcal{D}_{\text{c},i}$ and form the communication dataset $\mathcal{D}_{\text{c}}$. The rest is a direct application of DEC-apx-GP with the local augmented datatset $\mathcal{D}_{+i}$ for all $i \in \mathcal{V}$. Algorithm 4 presents the implementation details of DEC-gapx-GP. We show the structure of the proposed method in Fig. 2-(b). The communication dataset $\mathcal{D}_{\text{c}}$ is illustrated in gray for every agent. The larger rectangular blocks represent the double size of local augmented datasets $\mathcal{D}_{+i} \in \mathbb{R}^{2N_i}$ when compared to the local datasets $\mathcal{D}_1, \dots, \mathcal{D}_M$. Larger circular objects indicate that the augmented covariance matrices $C_{\theta,+i} \in \mathbb{R}^{2N_i \times 2N_i}$ of DEC-gapx-GP have double dimension, when compared to the local covariance matrices $C_{\theta,i} \in \mathbb{R}^{N_i \times N_i}$ for all $i \in \mathcal{V}$ of DEC-c-GP and DEC-apx-GP. Note that DEC-gapx-GP inherits the properties of DEC-apx-GP (Theorem IV-B).

D. TIME, SPACE, AND COMMUNICATION COMPLEXITY

Let the total number of iterations for the nested optimization problem (15b) be $s^{\text{end}}_{\text{nest}}$. The time complexity of every agent i is dominated by the inverse of the local covariance matrix $C^{-1}_{\theta,i}$ for every iteration of the nested optimization problem (15b), which results in $\mathcal{O}(s^{\text{end}}_{\text{nest}} N_i^3) = \mathcal{O}(s^{\text{end}}_{\text{nest}}(N^3/M^3))$ computations. The gradient for the nested optimization is provided in Appendix B. Moreover, every agent i occupies $\mathcal{O}(N_i^2 + DN_i + (D+2) + (\text{card}(\mathcal{N}_i)+1)(D+2)) = \mathcal{O}(N^2/M^2 + D(N/M) + (\text{card}(\mathcal{N}_i)+2)(D+2))$ memory to store $C^{-1}_{\theta,i}$, $\mathcal{D}_i$, $p_i^{(s)}$, $\boldsymbol{\theta}_i^{(s)}$, and $\{\boldsymbol{\theta}_j^{(s)}\}_{j\in\mathcal{N}_i}$. The total number of communications for each agent is $\mathcal{O}(s^{\text{end}}_{\text{DEC-c-GP}}(D+2))$ to transmit the hyperparameters to its neighbors. The complexity of DEC-c-GP is presented in Table 3 along with the other two proposed decentralized GP training methods.

The local time complexity of DEC-apx-GP is reduced to $\mathcal{O}(N_i^3) = \mathcal{O}(N^3/M^3)$ for the inversion of the local covariance matrix $C^{-1}_{\theta,i}$ just once at every ADMM iteration. The space complexity is identical to DEC-c-GP and the total communications entail $\mathcal{O}(s^{\text{end}}_{\text{DEC-apx-GP}}(D+2))$ messages. The complexity of DEC-apx-GP is provided in Table 3 along with DEC-c-GP and another decentralized GP training method that is presented in the following Section.

The local time complexity of DEC-gapx-GP entails $\mathcal{O}((2N_i)^3) = \mathcal{O}(8(N^3/M^3))$ computations to invert the local augmented covariance $C_{\theta,+i} = K_{+i} + \sigma_\epsilon^2 I_{2N_i} \in \mathbb{R}^{2N_i \times 2N_i}$. The proposed method requires $\mathcal{O}((2N_i)^2 + D(2N_i) + (\text{card}(\mathcal{N}_i) + 2)(D+2)) = \mathcal{O}(4(N^2/M^2) + 2D(N/M) + (\text{card}(\mathcal{N}_i) + 2)(D+2))$ space to store $C^{-1}_{\theta,+i}$, $\mathcal{D}_{+i}$, $p_i^{(s)}$, $\boldsymbol{\theta}_i^{(s)}$, and $\{\boldsymbol{\theta}_j^{(s)}\}_{j\in\mathcal{N}_i}$. The total communication overhead is $\mathcal{O}(s^{\text{end}}_{\text{DEC-gapx-GP}}(D+2))$.

In Table 3, we list the time, space, and communication complexities for the proposed decentralized factorized GP training methods. The DEC-c-GP is the most computationally

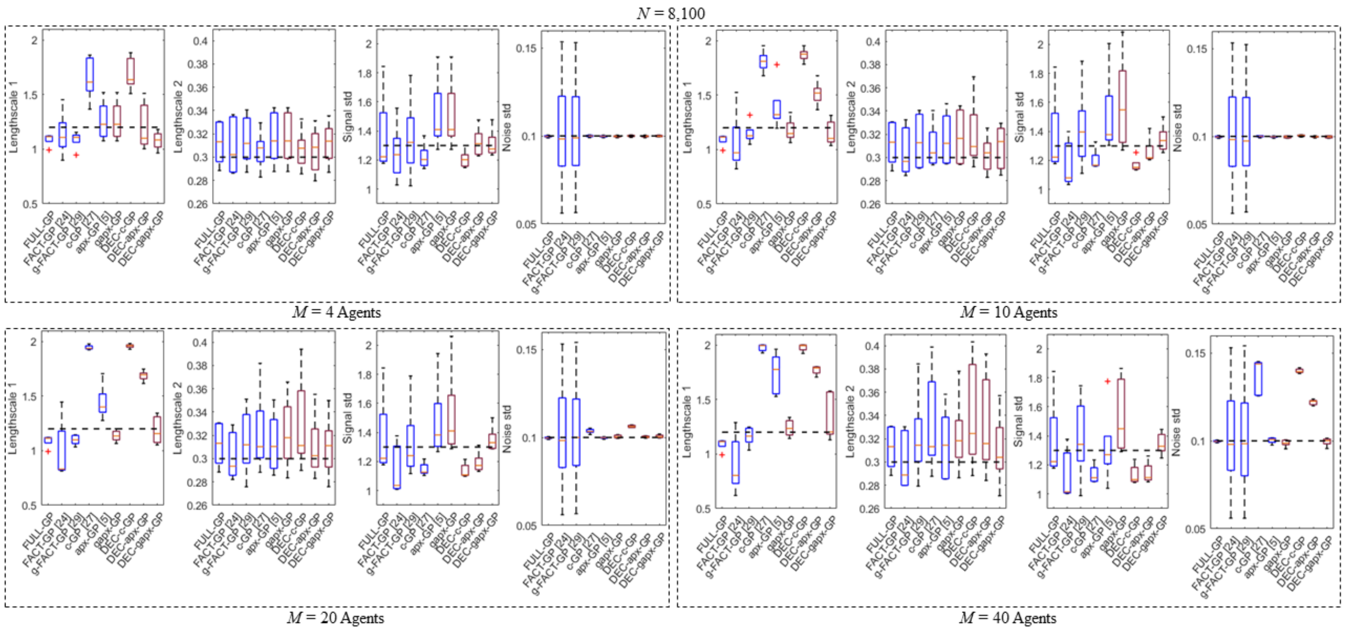

FIGURE 4. Accuracy of GP hyperparameter training using $N = 8,100$ data for four fleet sizes and 50 replications. The true values are demonstrated with a black dotted line. The existing GP training methods are shown in blue boxes. and the proposed in maroon coloured boxes.

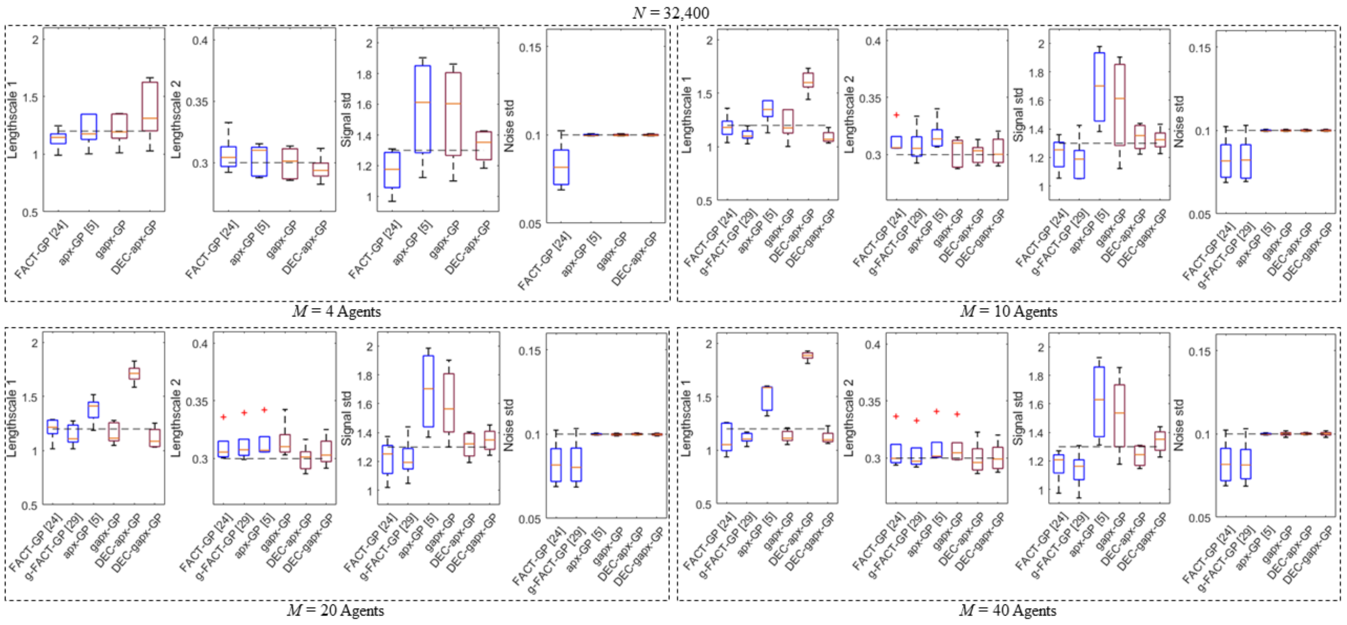

FIGURE 5. Accuracy of GP hyperparameter training using $N = 32,400$ data for four fleet sizes and 50 replications. The true values are demonstrated with a black dotted line. The existing GP training methods are shown in blue boxes and the proposed in maroon boxes.

expensive method, but it requires less communications than the other methods to converge. Therefore, the DEC-c-GP method favors applications with significant computational resources on the local nodes. Note that this method can also be extended with local augmented dataset $\mathcal{D}_{+i}$ for all $i \in \mathcal{V}$. Next, the DEC-apx-GP is the computationally most affordable method. The DEC-gapx-GP stands between the two former methods on time complexity, but requires more space and it is designed for large fleet multi-agent systems.

V. NUMERICAL EXPERIMENTS

In this section, we perform numerical experiments to illustrate the efficiency of the proposed methods. Synthetic data with known hyperparameters values are employed to evaluate the GP training methods in four aspects: i) hyperparameter estimation accuracy; ii) computation time per agent; iii) communications per agent; and iv) comparison with centralized GP training techniques. All numerical experiments are conducted in MATLAB on an Intel Core i7-6700 CPU @3.40 GHz.

We conduct 2,000 numerical experiments where we gener-

TABLE 4. Time & Communication Rounds of GP Training Methods

		$N = 8,100$		$N = 32,400$	
M	Method	Time [s]	Comms s^{end}	Time [s]	Comms s^{end}
	FULL-GP	2,114.2	-	-	-
4	FACT-GP [24]	75.9	186.0	2,361.9	196.0
	g-FACT-GP [29]	332.1	160.0	>3,000	-
	c-GP [27]	404.1	141.4	-	-
	apx-GP [5]	**26.8**	43.6	**817.6**	45.2
	gapx-GP	67.3	**39.7**	2,074.2	**42.1**
	DEC-c-GP	414.1	100	-	-
	DEC-apx-GP	**61.9**	100	**1,821.3**	100
	DEC-gapx-GP	328.1	100	>3,000	-
10	FACT-GP [24]	9.8	179.6	228.2	194.2
	g-FACT-GP [29]	31.8	131.8	1,035.6	155.2
	c-GP [27]	92.1	193.8	-	-
	apx-GP [5]	**3.8**	47.8	**88.8**	46.8
	gapx-GP	15.1	**42.2**	522.2	**44.3**
	DEC-c-GP	82.4	100	-	-
	DEC-apx-GP	**8.4**	100	**188.8**	100
	DEC-gapx-GP	38.5	100	1,123.4	100
20	FACT-GP [24]	2.6	172.6	46.6	226.2
	g-FACT-GP [29]	7.0	127.2	199.4	167.6
	c-GP [27]	31.4	127.8	-	-
	apx-GP [5]	**1.3**	56.2	**18.3**	49.8
	gapx-GP	4.1	**50.6**	85.8	**45.6**
	DEC-c-GP	30.4	100	-	-
	DEC-apx-GP	**2.2**	100	**36.9**	100
	DEC-gapx-GP	8.1	100	185.8	100
40	FACT-GP [24]	0.5	139.6	9.1	160.0
	g-FACT-GP [29]	1.8	112.2	30.9	128.6
	c-GP [27]	8.9	66.6	-	-
	apx-GP [5]	**0.3**	56.4	**4.6**	54.4
	gapx-GP	1.2	**51.2**	17.9	**49.2**
	DEC-c-GP	9.1	100	-	-
	DEC-apx-GP	**0.5**	100	**8.2**	100
	DEC-gapx-GP	2.5	100	36.4	100

TABLE 5. Comparison of Centralized and Decentralized GP Training

M	Method	Hyperparameter Accuracy	Computational Scalability	Commu-nications
	FULL-GP	Good	Bad	-
4	FACT-GP [24]	Good	Good	Moderate
	g-FACT-GP [29]	Good	Moderate	Moderate
	c-GP [27]	Moderate	Moderate	Moderate
	apx-GP [5]	Good	**Best**	Good
	gapx-GP	**Best**	Good	**Best**
	DEC-c-GP	Moderate	Moderate	Moderate
	DEC-apx-GP	Good	**Best**	Moderate
	DEC-gapx-GP	**Best**	Moderate	Moderate
10	FACT-GP [24]	Good	Good	Bad
	g-FACT-GP [29]	**Best**	Moderate	Moderate
	c-GP [27]	Moderate	Bad	Bad
	apx-GP [5]	Good	**Best**	Good
	gapx-GP	**Best**	Good	**Best**
	DEC-c-GP	Bad	Moderate	Moderate
	DEC-apx-GP	Bad	**Best**	Moderate
	DEC-gapx-GP	**Best**	Good	Moderate
20	FACT-GP [24]	Good	**Best**	Bad
	g-FACT-GP [29]	**Best**	Good	Moderate
	c-GP [27]	Bad	Bad	Moderate
	apx-GP [5]	Moderate	**Best**	Good
	gapx-GP	**Best**	Good	**Best**
	DEC-c-GP	Bad	Moderate	Moderate
	DEC-apx-GP	Bad	**Best**	Moderate
	DEC-gapx-GP	**Best**	Good	Moderate
40	FACT-GP [24]	Moderate	**Best**	Moderate
	g-FACT-GP [29]	**Best**	Good	Moderate
	c-GP [27]	Bad	Moderate	Good
	apx-GP [5]	Bad	**Best**	Good
	gapx-GP	**Best**	Good	**Best**
	DEC-c-GP	Bad	Moderate	Moderate
	DEC-apx-GP	Bad	**Best**	Moderate
	DEC-gapx-GP	**Best**	Good	Moderate

ate datasets by using the observation model (1) and the separable squared exponential kernel (2) with hyperparameters $\boldsymbol{\theta} = (l_1, l_2, \sigma_f, \sigma_\epsilon)^\intercal = (1.2, 0.3, 1.3, 0.1)^\intercal$. In particular, we have two dataset sizes ($N = 8,100$ and $N = 32,400$) for five generative random functions and perform 50 replications on each function. An example of five generative GP functions for $N = 8,100$ data is presented in Fig. 3. Note that the smaller the length-scale l, the more wiggly is the random function. Since $l_2 < l_1$, the profile of the generative GP functions is more uneven along the y-axis rather than the x-axis. We equally partition the space of interest $\mathbb{S} = [0, 2]^2$ (Remark 1) along the x-axis according to fleet sizes $M = \{4, 10, 20, 40\}$, and assign local datasets that lie in the corresponding local space. We compare the global GP training FULL-GP; the centralized FACT-GP [24], g-FACT-GP [29], c-GP [27], and apx-GP [5]; to the proposed centralized (gapx-GP) and decentralized (DEC-c-GP, DEC-apx-GP, DEC-gapx-GP) methods. All decentralized GP training methods follow a path graph topology that is the most parsimonious connected network. Thus, we study the worst case scenario in terms of network connectivity (Remark 4). All methods start from the same initial vector value $(l_1^{(0)}, l_2^{(0)}, \sigma_f^{(0)}, \sigma_\epsilon^{(0)})^\intercal = (2, 0.5, 1, 1)^\intercal$. The penalty parameter of the augmented Lagrangian is set to $\rho = 500$, the decentralized ADMM tolerance $\text{TOL}_{\text{ADMM}} = 10^{-3}$, the positive Lipschitz constant of the approximation (12) $L_i = 5,000$, and the regulation positive constant of the approximation (16) $\kappa_i = 5,000$ for all $i \in \mathcal{V}$. For the nested optimization of c-GP (11b) and DEC-c-GP (15b) we use gradient descent with step size $\alpha = 10^{-5}$. All decentralized GP training methods terminate after $s^{\text{end}} = 100$ predetermined communication rounds, yielding identical communication complexity (Table 3). Any algorithm that takes over 3,000 s to be executed is terminated.

In Fig. 4, we show the boxplots of the estimated hyperparameters using $N = 8,100$ data. Blue boxes illustrate existing GP training methods and maroon boxes represent the proposed GP training methods. The corresponding average computation time per agent and the communication rounds are shown in Table 4. Provided the communication rounds s^{end}, the communication complexity can be computed according to Table 1, 2. For the case of $M = 4$ agents, all centralized methods provide accurate hyperparameters estimates except of the c-GP on l_1. In terms of computation time, c-GP is the more demanding method, whereas FACT-GP, apx-GP, and gapx-GP converge very fast, outperforming FULL-GP by two orders of magnitude for similar or even better level of accuracy. The least communication rounds are achieved by the proposed methodology gapx-GP which results in the lowest communication complexity. Regarding the decentralized methods, both DEC-apx-GP and DEC-gapx-GP produce accurate hyperparameter estimates, whereas DEC-c-GP is inaccurate on l_1. DEC-apx-GP requires less computation time per agent than the other two decentralized methods. As we increase the number of agents ($M = 10$ and $M = 20$ agents), the hyperparameter estimation accuracy deteriorates for all centralized methods except of the proposed gapx-GP. In addition, gapx-GP results in the lowest communication complexity and in competitive computation time per agents, outperformed only by apx-GP. Regarding the decentralized GP training methods, the hyperparameter estimation of DEC-gapx-GP is the most accurate. Both DEC-apx-GP and DEC-c-GP provide reasonable estimates for all hyperparameters other than l_1. The lowest computation per entity is measured for DEC-apx-GP, while the most accurate method DEC-gapx-GP requires four times more computations than DEC-apx-GP. For $M = 40$ agents, the proposed gapx-GP produces the most accurate hyperparameter estimates with only g-FACT-GP competing. However, g-FACT-GP requires more computation time per agent and exchanges double the amount of messages to converge than the proposed gapx-GP. From the proposed decentralized methods, DEC-gapx-GP is accurate (Remark 5) for larger fleet sizes and requires reasonable local computations (Table 4).

We present the boxplots of the estimated hyperparameters using $N = 32,400$ data in Fig. 5, and in Table 4 we list the computation time per agent as well as the communication rounds. The FULL-GP, c-GP, and DEC-c-GP methods are not implemented for $N = 32,400$ data, as we expect significantly high computation time (Remark 3). For $M = 4$ agents, both g-FACT-GP and DEC-gapx-GP exceeded the time limit of 3,000 s for convergence. Among the feasible centralized methods for $N = 32,400$ data, apx-GP and gapx-GP are more accurate than FACT-GP. All methods are computationally expensive as each agent i is assigned with $N_i = 32,400/4 = 8,100$ data, yet apx-GP is the fastest. Regarding the decentralized methods, DEC-apx-GP is the only feasible method and produces accurate hyperparameter estimates. As we increase the number of agents ($M = 10$ and $M = 20$ agents), the number of data is distributed to local agents, and thus g-FACT-GP and DEC-gapx-GP can be implemented. Since the number of data is high, all centralized methods produce accurate hyperparameters estimates. Yet, apx-GP is computationally more efficient. Although the proposed gapx-GP requires more time to converge, the communication overhead is the least. Among the decentralized methods, DEC-gapx-GP is more accurate, but computationally more demanding than DEC-apx-GP. For the case of $M = 40$ agents, the most accurate centralized hyperparameter estimator is the gapx-GP with the lowest information exchange requirements. The fastest centralized method is the apx-GP, yet its accuracy is moderate. Regarding the decentralized methods, DEC-gapx-GP remains accurate and requires reasonable computation time.

In Table 5, we compare qualitatively all methods, where the proposed methods are shown in magenta font. Overall, for $N = 8,100$ the proposed gapx-GP is the most accurate centralized GP training method, especially as the fleet size increases. Moreover, gapx-GP requires reasonable computations and it is the most efficient method with respect to communication. Among the proposed decentralized GP training methods, DEC-gapx-GP is the most accurate method, yet DEC-apx-GP produces competitive hyperparameter estimates for medium and small fleet size. DEC-apx-GP is the fastest decentralized GP training method, while DEC-gapx-GP is more demanding with reasonable computational resources. In principle, as we increase the number of agents, the computation is distributed and thus yields lower computation time per agent. Note that the hyperparameter estimation accuracy improves as we obtain more data which leads to higher accuracy for $N = 32,400$ data. Some techniques are not scalable for the larger dataset $N = 32,400$, especially when the fleet size is small $M = 4$. However, for larger fleet size the distribution of data facilitates the execution of most methods. Among the centralized methods, apx-GP is accurate and requires significantly less computational time for small fleet size, but as we increase the number of agents the proposed gapx-GP becomes computationally more efficient and remains accurate. Similarly, DEC-apx-GP is accurate and computationally less demanding for small fleet size, but DEC-gapx-GP becomes more computationally efficient as we distribute the data to more agents.

VI. CONCLUSION AND FUTURE WORK

This paper proposes decentralized methods to implement GP training in networks that cover a broad spectrum of multi-agent learning applications. The proposed methods can be employed for various fleet sizes with different computation and communication capabilities. We use distributed optimization methods of ADMM to aggregate local GP models. A closed-form solution of the decentralized ADMM is derived for the case of GP hyperparameter training with maximum likelihood estimation. DEC-apx-GP is shown to achieve competitive accuracy in hyperparameter estimates for small and medium fleet sizes, whereas DEC-gapx-GP produces accurate hyperparameter estimates for all fleet sizes with reasonable computations of local entities. Additionally, we propose a centralized GP training method, the gapx-GP, that improves the accuracy of hyperparameter estimates for medium and large fleet sizes, entails reasonable computations, and requires little information exchange.

A. PARTIAL DERIVATIVE OF SSE COVARIANCE FUNCTION

The partial derivative of the covariance matrix in (4) is computed with respect to each hyperparameter as $\partial C_\theta/\partial \boldsymbol{\theta} = (\partial C_\theta/\partial l_1, \partial C_\theta/\partial l_2, \dots, \partial C_\theta/\partial l_D, \partial C_\theta/\partial \sigma_f, \partial C_\theta/\partial \sigma_\epsilon)^\intercal \in \mathbb{R}^{(D+2)N\times N}$. Using the SSE kernel (2) we obtain,

$$\left[\frac{\partial C_\theta}{\partial l_d}\right]_{ij} = \sigma_f^2 \left[\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_{di}-x_{dj})^2}{l_d^2}\right\}\frac{(x_{di}-x_{dj})^2}{l_d^3}\right]_{ij} = \frac{[K]_{ij}\left[(x_{di}-x_{dj})^2\right]_{ij}}{l_d^3},$$

where $\partial C_\theta/\partial l_d \in \mathbb{R}^{N\times N}$. For the signal variance we get,

$$\left[\frac{\partial C_\theta}{\partial \sigma_f}\right]_{ij} = 2\sigma_f \left[\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_{di}-x_{dj})^2}{l_d^2}\right\}\right]_{ij} = \frac{2[K]_{ij}}{\sigma_f},$$

where $\partial C_\theta/\partial \sigma_f \in \mathbb{R}^{N\times N}$. Note that we express the partial derivatives as functions of the correlation matrix K, because it has already been computed to construct the covariance matrix, i.e., $C_\theta = K + \sigma_\epsilon^2 I_N$. Lastly, for the noise variance $\partial C_\theta/\partial\sigma_\epsilon = 2\sigma_\epsilon I_N \in \mathbb{R}^{N\times N}$.

B. GRADIENT FOR NESTED PROBLEM OF DEC-C-GP

Let the objective for the nested optimization problem (15b) of the DEC-c-GP to be $\mathcal{K}_i = \mathcal{L}_i(\boldsymbol{\theta}_i) + \boldsymbol{\theta}_i^\intercal p_i^{(s+1)} + \rho\sum_{j\in\mathcal{N}_i}\|\boldsymbol{\theta}_i - (\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)})/2\|_2^2$, then its gradient yields,

$$\frac{\partial \mathcal{K}_i}{\partial \boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}}\mathcal{L}_i(\boldsymbol{\theta}_i) + p_i^{(s+1)} + 2\rho\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_i - \frac{\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)}}{2}.$$

The gradient of the local log-likelihood $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$ yields,

$$\frac{\partial \mathcal{L}_i(\boldsymbol{\theta}_i)}{\partial \boldsymbol{\theta}} = \frac{1}{2}\operatorname{tr}\left\{\left(C_{\theta,i}^{-1} - C_{\theta,i}^{-1}y_i y_i^\intercal C_{\theta,i}^{-1}\right)\frac{\partial C_{\theta,i}}{\partial \boldsymbol{\theta}}\right\}, \tag{20}$$

where $\partial C_{\theta,i}/\partial\boldsymbol{\theta}$ is derived in Appendix A for the SSE covariance function (2).

C. PROOF OF THEOREM 1

Let us employ the local objective of (17b) as,

$$\mathcal{Q}_i(\boldsymbol{\theta}_i) = \nabla_{\boldsymbol{\theta}}^\intercal\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right)\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right) + \frac{\kappa_i}{2}\left\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right\|_2^2 + \boldsymbol{\theta}_i^\intercal p_i^{(s+1)} + \rho\sum_{j\in\mathcal{N}_i}\left\|\boldsymbol{\theta}_i - \frac{\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)}}{2}\right\|_2^2.$$

where $\mathcal{Q}_i : \mathbb{R}^{D+2} \to \mathbb{R}$. Factor out the optimizing parameter $\boldsymbol{\theta}_i$ to obtain,

$$\begin{aligned}\mathcal{Q}_i(\boldsymbol{\theta}_i) &= \nabla_{\boldsymbol{\theta}}^\intercal\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right)\boldsymbol{\theta}_i - c_1 + \frac{\kappa_i}{2}\left(\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i - 2\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i^{(s)} + c_2\right) + \boldsymbol{\theta}_i^\intercal p_i^{(s+1)} + \mathcal{T}_i \\ &= \boldsymbol{\theta}_i^\intercal\left(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) - \kappa_i\boldsymbol{\theta}_i^{(s)} + p_i^{(s+1)}\right) + \frac{\kappa_i}{2}\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i + \mathcal{T}_i,\end{aligned} \tag{21}$$

where $\mathcal{T}_i = \rho\sum_{j\in\mathcal{N}_i}\|\boldsymbol{\theta}_i - (\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)})/2\|_2^2$, $c_1 = -\nabla_{\boldsymbol{\theta}}^\intercal\mathcal{L}_i(\boldsymbol{\theta}_i^{(s)})\boldsymbol{\theta}_i^{(s)}$, and $c_2 = \boldsymbol{\theta}_i^{\intercal(s)}\boldsymbol{\theta}_i^{(s)}$. Note that c_1, c_2 are constants with respect to $\boldsymbol{\theta}_i$ and thus irrelevant to the optimization (14). For any strongly connected graph topology, $\mathcal{T}_i$ can be expressed as,

$$\begin{aligned}\mathcal{T}_i &= \rho\sum_{j\in\mathcal{N}_i}\left\|\boldsymbol{\theta}_i - \frac{\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)}}{2}\right\|_2^2 \\ &= \rho\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^\intercal\left(\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)}\right) + c_3 \\ &= \rho\operatorname{card}(\mathcal{N}_i)\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i - \rho\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_j^{(s)} \\ &= \rho\operatorname{card}(\mathcal{N}_i)\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i - \rho\operatorname{card}(\mathcal{N}_i)\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i^{(s)} - \rho\boldsymbol{\theta}_i^\intercal\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_j^{(s)} \\ &= \operatorname{card}(\mathcal{N}_i)\rho\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i - \rho\boldsymbol{\theta}_i^\intercal\left(\operatorname{card}(\mathcal{N}_i)\boldsymbol{\theta}_i^{(s)} + \sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_j^{(s)}\right),\end{aligned} \tag{22}$$

where $c_3 = (1/4)(\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)})^\intercal(\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)})$ is a constant and thus ignored. The local objective $\mathcal{Q}_i$ results in,

$$\begin{aligned}\mathcal{Q}_i(\boldsymbol{\theta}_i) &= \boldsymbol{\theta}_i^\intercal\left(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) - \kappa_i\boldsymbol{\theta}_i^{(s)} + p_i^{(s+1)}\right) + \frac{\kappa_i}{2}\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i \\ &\quad + \operatorname{card}(\mathcal{N}_i)\rho\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i - \rho\boldsymbol{\theta}_i^\intercal\left(\operatorname{card}(\mathcal{N}_i)\boldsymbol{\theta}_i^{(s)} + \sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_j^{(s)}\right) \\ &= \boldsymbol{\theta}_i^\intercal\left(\nabla_{\boldsymbol{\theta}}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) - \Big(\kappa_i + \operatorname{card}(\mathcal{N}_i)\rho\Big)\boldsymbol{\theta}_i^{(s)} + p_i^{(s+1)} - \rho\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_j^{(s)}\right) + \left(\frac{\kappa_i}{2} + \operatorname{card}(\mathcal{N}_i)\rho\right)\boldsymbol{\theta}_i^\intercal\boldsymbol{\theta}_i.\end{aligned} \tag{23}$$

Next, we show that the local objective $\mathcal{Q}_i$ (23) is a convex function in a quadratic form [49] by computing its Hessian,

$$\mathcal{H}_{\mathcal{Q}_i} = \frac{\partial^2\mathcal{Q}_i}{\partial\boldsymbol{\theta}_i^2} = \left(\kappa_i + 2\operatorname{card}(\mathcal{N}_i)\rho\right)I_{D+2} \succ 0.$$

Since the local objective $\mathcal{Q}_i$ is convex and quadratic, we can obtain a closed-form solution by computing the first derivative,

$$\frac{\partial\mathcal{Q}_i}{\partial\boldsymbol{\theta}_i} = \nabla_{\boldsymbol{\theta}}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) - \Big(\kappa_i + \operatorname{card}(\mathcal{N}_i)\rho\Big)\boldsymbol{\theta}_i^{(s)} + p_i^{(s+1)} - \rho\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_j^{(s)} + 2\Big(\frac{\kappa_i}{2} + \operatorname{card}(\mathcal{N}_i)\rho\Big)\boldsymbol{\theta}_i,$$

and then setting $\partial\mathcal{Q}_i/\partial\boldsymbol{\theta}_i = \mathbf{0}$ to obtain,

$$\boldsymbol{\theta}_i = \frac{1}{\kappa_i + 2\operatorname{card}(\mathcal{N}_i)\rho}\left(\rho\sum_{j\in\mathcal{N}_i}\boldsymbol{\theta}_j^{(s)} - \nabla_{\boldsymbol{\theta}}\mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) + \left(\kappa_i + \operatorname{card}(\mathcal{N}_i)\rho\right)\boldsymbol{\theta}_i^{(s)} - p_i^{(s+1)}\right).$$

The rest proof is a direct consequence of [47, Theorem 1].

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *arXiv preprint arXiv:1610.05492,* 2016.

[2] E. Horvitz and D. Mulligan, "Data, privacy, and the greater good," *Science*, vol. 349, no. 6245, pp. 253–255, 2015.

[3] J. Gielis, A. Shankar, and A. Prorok, "A critical review of communications in multi-robot systems," *Current Robotics Reports*, pp. 1–13, 2022.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Foundations and Trends in Machine Learning, 2011, vol. 3, no. 1.

[5] A. Xie, F. Yin, Y. Xu, B. Ai, T. Chen, and S. Cui, "Distributed Gaussian processes hyperparameter optimization for big data using proximal ADMM," *IEEE Signal Processing Letters*, vol. 26, no. 8, pp. 1197–1201, 2019.

[6] C.E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning, 2*nd ed. Cambridge, MA, USA: MIT Press, 2006.

[7] R. B. Gramacy, *Surrogates: Gaussian process modeling, design and optimization for the applied sciences*. Boca Raton, FL, USA: Chapman Hall/CRC, 2020.

[8] Y. Xu, J. Choi, and S. Oh, "Mobile sensor network navigation using Gaussian processes with truncated observations," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1118–1131, 2011.

[9] D. Gu and H. Hu, "Spatial Gaussian process regression with mobile sensor networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1279–1290, 2012.

[10] J. Chen, K. H. Low, Y. Yao, and P. Jaillet, "Gaussian process decentralized data fusion and active sensing for spatiotemporal traffic modeling and prediction in mobility-on-demand systems," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 901–921, 2015.

[11] W. Luo and K. Sycara, "Adaptive sampling and online learning in multi-robot sensor coverage with mixture of Gaussian processes," in *IEEE International Conference on Robotics and Automation,* 2018, pp. 6359–6364.

[12] T. N. Hoang, Q. M. Hoang, K. H. Low, and J. How, "Collective online learning of Gaussian processes in massive multi-agent systems," in *AAAI Conference on Artificial Intelligence,* vol. 33, no. 01, 2019, pp. 7850–7857.

[13] M. Tavassolipour, S. A. Motahari, and M. T. M. Shalmani, "Learning of Gaussian processes in distributed and communication limited systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 1928–1941, 2020.

[14] G. P. Kontoudis and D. J. Stilwell, "Prediction of acoustic communication performance in marine robots Using model-based kriging," in *American Control Conference,* 2021, pp. 3779-3786.

[15] G. P. Kontoudis and D. J. Stilwell, "Model-based learning of underwater acoustic communication performance for marine robots," *Robotics and Autonomous Systems*, vol. 142, pp. 103811, 2021.

[16] V. Suryan and P. Tokekar, "Learning a spatial field in minimum time with a team of robots," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1562–1576, 2020.

[17] G. P. Kontoudis and D. J. Stilwell, "Decentralized nested Gaussian processes for multi-robot systems," in *IEEE International Conference on Robotics and Automation,* 2021, pp. 8881–8887.

[18] M. Santos, U. Madhushani, A. Benevento, and N. E. Leonard, "Multi-robot learning and coverage of unknown spatial fields," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems,* 2021, pp. 137–145.

[19] F. Bullo, J. Cortes, and S. Martinez, *Distributed control of robotic networks: A mathematical approach to motion coordination algorithms*. Princeton University Press, 2009, vol.27.

[20] G. P. Kontoudis and D. J. Stilwell, "A comparison of kriging and cokriging for estimation of underwater acoustic communication performance," in *International Conference on Underwater Networks & Systems,* 2019, pp. 1–8.

[21] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[22] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[23] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in NeurIPS,* 2006, pp. 1257–1264.

[24] M. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in *International Conference on Machine Learning,* 2015, pp. 1481–1490.

[25] T. Halsted, O. Shorinwa, J. Yu and M. Schwager, "A Survey of Distributed Optimization Methods for Multi-Robot Systems," in *arXiv preprint arXiv:2103.12840,* 2021.

[26] V.-A. Le, L. Nguyen and T. X. Nghiem, "ADMM-based adaptive sampling strategy for nonholonomic mobile robotic sensor networks," *IEEE Sensors Journal*, vol. 21, no. 13, pp. 15369–15378, 2021.

[27] Y. Xu, F. Yin, W. Xu, J. Lin, and S. Cui, "Wireless traffic prediction with scalable Gaussian process: Framework, algorithms, and verification," *IEEE Journal on Selected Areas in Communication*, vol. 37, no. 6, pp. 1291–1306, 2019.

[28] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[29] H. Liu, J. Cai, Y. Wang, and Y. S. Ong, "Generalized robust Bayesian committee machine for large-scale Gaussian process regression," in *International Conference on Machine Learning,* 2018, pp. 3131–3140.

[30] P. Moreno-Muñoz, A. Artés, and M. Alvarez, "Modular Gaussian processes for transfer learning," in *Advances in Neural Information Processing Systems,* vol. 34, 2021, pp. 24730–24740.

[31] X. Yue and R. Al Kontar, "Federated Gaussian process: Convergence, automatic personalization and multi-fidelity modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[32] S. Chung and R. Al Kontar, "Federated multi-output Gaussian processes," *Technometrics*, vol. 66, no. 1, pp. 90–103, 2024.

[33] G. P. Kontoudis and D. J. Stilwell, "Decentralized Federated Learning using Gaussian Processes," in *International Symposium on Multi-Robot and Multi-Agent Systems,* 2023, pp. 1–7.

[34] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Effective graph resistance," *Linear algebra and its applications*, vol. 435, no. 10, pp. 2491–2506, 2011.

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

[36] J. W. Ng and M. P. Deisenroth, "Hierarchical mixture-of-experts model for large-scale Gaussian process regression," in *arXiv preprint arXiv:1412.3078,* 2014.

[37] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K. H. Johansson, "A survey of distributed optimization," *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[38] Y. Zheng and Q. Liu, "A review of distributed optimization: Problems, models and algorithms," *Neurocomputing*, vol. 483, pp. 446–459, 2022.

[39] D. J. C. Mackay, "Introduction to Gaussian processes," *NATO ASI series. Series F: Computer and System Sciences*, pp. 133–165, 1998.

[40] F. Pérez-Cruz, S. Van Vaerenbergh, J. J. Murillo-Fuentes, M. Lázaro-Gredilla, and I. Santamaria, "Gaussian processes for nonlinear signal processing: An overview of recent advances," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 40–50, 2013.

[41] Z. Chen and B. Wang, "How priors of initial hyperparameters affect Gaussian process regression models," *Neurocomputing*, vol. 275, pp. 1702–1710, 2018.

[42] S. Basak, S. Petit, J. Bect, and E. Vazquez, "Numerical issues in maximum likelihood parameter estimation for Gaussian process interpolation," in *International Conference on Machine Learning, Optimization and Data Science,* 2021.

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

[44] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5082–5095, 2017.

[45] A. Elgabli, J. Park, A. S. Bedi, M Bennis, and V. Aggarwal, "GADMM: Fast and Communication Efficient Framework for Distributed Machine Learning," *Journal of Machine Learning Research*, vol. 21, no. 76, pp. 1–39, 2020.

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

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

[48] D. M. Topkis, "Concurrent broadcast for information dissemination," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 10, pp. 1107–1112, 1985.

[49] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

GEORGE P. KONTOUDIS (M'22) received the M.S. and PhD degrees in Mechaincal Engineering and Electrical Engineering at Virginia Tech, in 2018 and 2021 respectively.

From January 2022 to December 2023, he was a Postdoctoral Research Associate in the Aerospace Engineering Department at the University of Maryland, College Park. Since 2024, he is an Assistant Professor with the Mechanical Engineering Department, Colorado School of Mines. His research interests include multi-agent systems, Gaussian processes, motion planning, and optimal control.

DANIEL J. STILWELL is a professor at the Bradley Department of Electrical & Computer Engineering at Virginia Tech. He obtained a B.S. in Electrical Engineering from the University of Massachusetts at Amherst, and M.S. and Ph.D. degrees in Electrical Engineering from Johns Hopkins University. His research interests include robotics, control theory, and estimation.