Decentralized Federated Learning using Gaussian Processes

George P. Kontoudis and Daniel J. Stilwell

***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 a scalable optimization algorithm for decentralized learning of GP hyperparameters in multi-agent systems. To distribute the implementation of GP training, we employ the alternating direction method of multipliers (ADMM). We provide a closed-form solution of the nested optimization of decentralized proximal ADMM for the case of GP modeling with the separable squared exponential kernel. Decentralized federated learning is promoted by prohibiting local data exchange between agents. The efficiency of the proposed method is illustrated with numerical experiments.**

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. Yet, gathering data to a central server may lead to network traffic congestion and security/privacy issues. To ensure data privacy, a promising solution is federated learning (FL). FL aims to implement ML techniques in centralized or decentralized networks, but with no communication of real data. 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. Such cases include autonomous vehicles and multi-robot systems in underwater, underground. and extreme weather applications. Our aim in this work is to develop a fully decentralized algorithm for approximate Gaussian process (GP) training that relaxes the computation and communication requirements with no data sharing and achieves similar performance to centralized methods.

Gaussian processes [1] are used in various multi-agent applications [2]–[16], 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 $\mathcal{O}(N^2)$ 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 multi-agent systems is the communication [17]. 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, which may vary in space and time [18].

Two major directions for GP approximations are based on global and local approaches [19]. Global approximation methods promote sparsity by using either a subset of N_{sub} observations or by constructing a set of N_{sub} pseudo-inputs, where $N_{\text{sub}} \ll N$ to perform GP training with a much smaller dataset [20], [21]. Sparse GPs have been used in mobile sensor networks to model spatial fields [3]. In [2], a GP with truncated observations in a mobile sensor network is proposed, and in [4] a subset of observations is used for traffic modeling. These methods require global knowledge of data, which increases inter-agent communications.

The second direction uses local approximation methods to reduce the computational burden of GP training [19]. A local approximate method with maximum likelihood estimation (MLE) is the factorized GP training [22]. That is a centralized method which is based on a server-client structure and distributes the computations to multiple entities. The main idea is to assume independence between sub-models, 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. Recently, Xu *et al.* [23] reformulated the factorized GP training method using the exact consensus alternating direction method of multipliers (ADMM) [24]. Consensus ADMM reduces the communication overhead of GP training, but requires high computational resources to solve a nested optimization problem at every ADMM itera-

G. P. Kontoudis is with the Mechanical Engineering Department, Colorado School of Mines, CO, USA. Email: george.kontoudis@mines.edu.

D. J. Stilwell is with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, VA, USA. Email: stilwell@vt.edu.

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

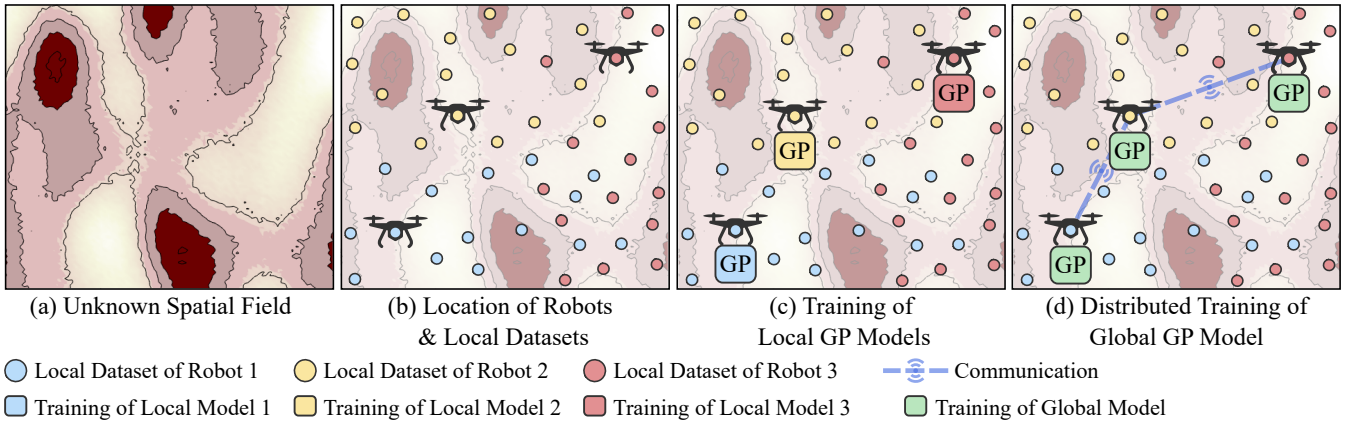

Fig. 1. Decentralized GP training in multi-agent systems using federated learning. (a) The exploration area of interest is an unknown spatial field. The unknown field can be of higher input dimension, i.e. $D > 2$, but for illustrative reasons we depict a 2D input environment. (b) The observation locations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

tion. Subsequently, the authors in [25] employed the inexact proximal ADMM [26] to alleviate the computation demand. However, both ADMM-based factorized GP training methods require a centralized network topology. In [27], the authors introduced an efficient centralized methodology, termed as generalized factorized GP training, that entails additional communication between agents to enrich local datasets with a global random dataset. Multiple studies revealed that ADMM is appealing in centralized multi-robot cases [28], [29]. Alternatively, many methods employ active learning of GP surrogates [30]–[36], to sequentially build small yet efficient datasets, but our focus is on local GP approximations.

The contributions of this paper are: i) the formulation of a decentralized GP training method (DEC-apx-GP) for connected graph topologies with no data exchange; and ii) the derivation of a closed-form solution for the nested optimization problem of the GP hyperparameter optimization. The proposed method achieves similar GP hyperparameter estimation accuracy with centralized GP training methods [22], [25]. The closed-form solution of the nested optimization problem enables scalable computations for GP training. Since the majority of multi-robot systems cannot rely on centralized methods for long-term operation, we envision that our method can effectively be used to address model learning problems with network topology challenges.

II. Preliminaries and Problem Statement

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

A. Foundations

The notation here is standard. The vector of n zeros is represented as $\mathbf{0}_n$ and the matrix of $n \times m$ zeros as $\mathbf{0}_{n\ m}$. The superscript in parentheses $y^{(s)}$ 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 $|\cdot|$, and the L_2 norm is denoted $\|\cdot\|_2$. The notation $\overline{\lambda}(\boldsymbol{F})$ and $\underline{\lambda}(\boldsymbol{F})$ denote the maximum and minimum eigenvalue of matrix $\boldsymbol{F}$ respectively. The i-th element of a vector $\boldsymbol{x}$ is denoted x_i and $\boldsymbol{x}_i$ denotes the vector $\boldsymbol{x}$ of agent i. A collection of elements that comprise a vector $\boldsymbol{x} \in \mathsf{R}^N$ is denoted $\{x_i\}_{i=1}^N$.

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 $\mathcal{V} = v_1, \ldots, v_M$ 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 $\boldsymbol{A} = [a_{ij}] \in \mathsf{R}^{M\ \ M}$, 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 [37].

Assumption 1. *[38] 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,

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

where $\boldsymbol{x} \in \mathsf{R}^D$ is the input location with D the input dimension, $f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}^0))$ is a zero-mean GP with covariance function $k : \mathsf{R}^D \times \mathsf{R}^D \to \mathsf{R}$, 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,

$$k(\boldsymbol{x}, \boldsymbol{x}^0) = \sigma_f^2 \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x_d^0)^2}{l_d^2}\right\}, \tag{2}$$

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 the data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$, where $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ the inputs, $\boldsymbol{y} = \{y_n\}_{n=1}^N$ the outputs, and N the number of observations.

A GP is trained to find the hyperparameters $= (l_1, \ldots, l_D, \sigma_f, \sigma_\epsilon)^\intercal \in \Theta \subset \mathsf{R}^{D+2}$ that maximize the

marginal log-likelihood,

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

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

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

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

The bound constraints (3b) on the length-scales l_d ensure that the correlation matrix is positive semi-definite.

C. Factorized GP Training (FACT-GP)

Let each agent i to collect local observations and form the local dataset $\{\mathcal{D}_i = \{\boldsymbol{X}_i, \boldsymbol{y}_i\}\}_{i=1}^{M}$ 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 $i \neq j$. For privacy reasons, we presume that the local datasets $\mathcal{D}_i$ cannot be communicated to other agents. Factorized GP training (FACT-GP) [22], [39] considers a centralized topology, where every entity i communicates to a central node with significant computational and storage resources. The centralized topology arises several problems such as security, traffic network congestion, and privacy. In addition, for certain cases (e.g., autonomous vehicles and multi-robot systems), distant agents may be subject to communication range limitations.

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. *All local sub-models $\mathcal{M}_i$ are statistically independent.*

The factorized GP training relies on Assumption 3. This implies that the global marginal likelihood can be approximated by the product of local likelihoods, which leads to,

$$p(\boldsymbol{y} \mid \boldsymbol{X}) \approx \prod_{i=1}^{M} p_i(\boldsymbol{y}_i \mid \boldsymbol{X}_i), \tag{4}$$

where $p_i(\boldsymbol{y}_i \mid \boldsymbol{X}_i) \sim \mathcal{N}(0, \boldsymbol{C}_{\theta,i})$ is the local marginal likelihood of the i-th node with local covariance matrix $\boldsymbol{C}_{\theta,i} = \boldsymbol{K}_i + \sigma_{\epsilon}^2 I_{N_i}$. Moreover, the factorized approximation (4) yields a block diagonal approximation of the covariance matrix $\boldsymbol{C}_{\theta}^{-1} \approx \operatorname{diag}(\boldsymbol{C}_{\theta,1}^{-1}, \ldots, \boldsymbol{C}_{\theta,M}^{-1})$. Thus, the global marginal log-likelihood is approximated by $\mathcal{L} \approx \sum_{i=1}^{M} \mathcal{L}_i$ which results in,

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

with local marginal log-likelihood $\mathcal{L}_i = \log p_i(\boldsymbol{y}_i \mid \boldsymbol{X}_i)$,

$$\mathcal{L}_i = -\frac{1}{2}\Big(\boldsymbol{y}_i^{\intercal} \boldsymbol{C}_{\theta,i}^{-1} \boldsymbol{y}_i + \log|\boldsymbol{C}_{\theta,i}| + N_i \log 2\pi\Big). \tag{5}$$

The gradient of the global log-likelihood in FACT-GP is computed by $\nabla \mathcal{L} = \sum_{i=1}^{M} \nabla \mathcal{L}_i$ [23], [25]. The optimization uses the local negative log-likelihood,

$$\text{(P2)} \quad \hat{\theta} = \arg\min \quad \sum_{i=1}^{M} \boldsymbol{y}_i^{\intercal} \boldsymbol{C}_{\theta,i}^{-1} \boldsymbol{y}_i + \log|\boldsymbol{C}_{\theta,i}| \tag{6a}$$

$$\text{s.to} \quad \theta_i > \mathbf{0}_{D+2}, \quad \forall i \in \mathcal{V}, \tag{6b}$$

where $\theta_i = \{l_{1,i}, \ldots, l_{D,i}, \sigma_{f,i}\sigma_{\epsilon,i}\}$ the local hyperparameters of agent i. Similarly to (3), constraint (6b) imposes positivity on the hyperparameters for all agents $i \in \mathcal{V}$.

D. Problem Definition

Problem 1. *Under Assumption 2, solve the optimization problem (P2) in* (6) *to estimate the GP hyperparmeters $\hat{\theta}$ for a connected decentralized network topology (Assumption 1) with independent datasets for each agent (Assumption 3).*

III. Proposed Decentralized GP Training

In this section, we introduce a method to address Problem 1 based on the *edge formulation* of ADMM [40] that yields parallel updates and decentralizes the factorized GP training. Let us consider the edge formulation of ADMM to formulate the optimization problem. This is a variation of the consensus ADMM [24] for decentralized networks. If Assumption 1 holds, then the consensus ADMM problem is equivalent to the edge formulation of ADMM that yields,

$$\text{(P3)} \quad \hat{\theta} = \arg\min \quad \sum_{i=1}^{M} \boldsymbol{y}_i^{\intercal} \boldsymbol{C}_{\theta,i}^{-1} \boldsymbol{y}_i + \log|\boldsymbol{C}_{\theta,i}| \tag{7a}$$

$$\text{s.to} \quad \theta_i = \theta_{ij}, \quad \forall i \in \mathcal{V},\, j \in \mathcal{N}_i, \tag{7b}$$

$$\theta_j = \theta_{ij}, \quad \forall i \in \mathcal{V},\, j \in \mathcal{N}_i, \tag{7c}$$

where θ_{ij} are auxiliary variables. Constraints (7b) and (7c) imply that each agent i is allowed to have its own opinion for the hyperparameters θ_i, yet at the end of the optimization all agents in the neighborhood $\mathcal{N}_i$ must agree on the neighborhood values θ_{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 [41]. In addition, the group ADMM [42] offers a decentralized optimization method, yet for a specific graph topology. Thus, we find that the edge formulation is more suitable for decentralized GP training.

The formulation of the decentralized consensus ADMM is discussed in [43]. After rendering the augmented Lagrangian for (P3) in (7) we obtain the decentralized consensus ADMM iterative scheme,

$$\boldsymbol{p}_i^{(s+1)} = \boldsymbol{p}_i^{(s)} + \rho \sum_{j \in \mathcal{N}_i} \left(\theta_i^{(s)} - \theta_j^{(s)}\right), \tag{8a}$$

$$\theta_i^{(s+1)} = \arg\min_{\theta_i} \Big\{ \mathcal{L}_i(\theta_i) + \theta_i^{\intercal} \boldsymbol{p}_i^{(s+1)} + \rho \sum_{j \in \mathcal{N}_i} \Big\| \theta_i - \frac{\theta_i^{(s)} + \theta_j^{(s)}}{2} \Big\|_2^2 \Big\}, \tag{8b}$$

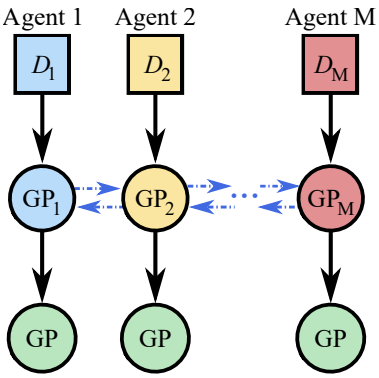

Fig. 2. The structure of the proposed decentralized GP training method (DEC-apx-GP). Blue dotted lines correspond to communication. Every agent i has access to its local dataset $\mathcal{D}_i$. The agents are allowed to have their own opinion on the GP hyperparameter $\boldsymbol{\theta}_i$ using exclusively $\mathcal{D}_i$, but after communicating they all agree on the same GP hyperparameter vector $\boldsymbol{\theta}$.

where $\rho > 0$ is the penalty term of the augmented Lagrangian and $\boldsymbol{p}_i^{(s)} = \sum_{j \in \mathcal{N}_i} (\boldsymbol{u}_{ij}^{(s)} + \boldsymbol{v}_{ij}^{(s)})$ is the sum of the dual variables $\boldsymbol{u}_{ij}^{(s)}$ and $\boldsymbol{v}_{ij}^{(s)}$ corresponding to constraints (7b) and (7c). Note that (8a) imposes initial values $\boldsymbol{p}_i^{(0)} = \mathbf{0}$.

Remark 1. *A disadvantage of* (8) *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* (8b) *which entails the computation of* $\mathcal{L}_i$ (5) *that involves the inversion of the local covariance matrix* $\boldsymbol{C}_{\theta,i}^{-1}$.

To address the scalability of decentralized ADMM (8), we extend the decentralized inexact proximal consensus ADMM [44] with an analytical solution of the nested optimization. A proximal step is taken based on a first-order approximation of the local log-likelihood $\mathcal{L}_i$ (5) around $\boldsymbol{\theta}_i^{(s)}$,

$$\mathcal{L}_i(\boldsymbol{\theta}_i) \approx \nabla^\intercal \mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right)\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right) + \frac{\kappa_i}{2}\left\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right\|_2^2, \tag{9}$$

where $\kappa_i > 0$ is a penalty parameter of the proximal term for all $i \in \mathcal{V}$. After rendering the augmented Lagrangian for (P3) in (7) we obtain the decentralized inexact proximal consensus ADMM iterative scheme,

$$\boldsymbol{p}_i^{(s+1)} = \boldsymbol{p}_i^{(s)} + \rho \sum_{j \in \mathcal{N}_i}\left(\boldsymbol{\theta}_i^{(s)} - \boldsymbol{\theta}_j^{(s)}\right), \tag{10a}$$

$$\begin{aligned}\boldsymbol{\theta}_i^{(s+1)} = \arg\min_{\boldsymbol{\theta}_i} \Big\{ &\nabla^\intercal \mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right)\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right) + \\ &\frac{\kappa_i}{2}\left\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^{(s)}\right\|_2^2 + \boldsymbol{\theta}_i^\intercal \boldsymbol{p}_i^{(s+1)} + \\ &\rho \sum_{j \in \mathcal{N}_i}\left\|\boldsymbol{\theta}_i - \frac{\boldsymbol{\theta}_i^{(s)} + \boldsymbol{\theta}_j^{(s)}}{2}\right\|_2^2\Big\}. \end{aligned} \tag{10b}$$

The linearization (9) allows the evaluation of the local log-likelihood $\mathcal{L}_i$ (5) at a fixed point $\boldsymbol{\theta}_i^{(s)}$ and not at the optimizing variable $\boldsymbol{\theta}_i$. Thus, the nested optimization (10b) entails significantly less computations than (8b), because we need to compute $\nabla^\intercal \mathcal{L}_i(\boldsymbol{\theta}_i^{(s)})$ just once (10b) and not at every iteration of the nested optimization (8b) (Remark 1). Next, we provide our main Theorem that extends [44] by deriving a closed-form solution for the nested optimization (10b) to reduce the computations and improve the accuracy.

Theorem 1. *Consider a strongly connected decentralized network (Assumption 1) where the agents are not allowed to communicate their datasets (Assumption 2). Let Assumption 3 hold for the local sub-models* $\mathcal{M}_i$ *and allow the penalty term of the first-order approximation* κ_i *to be sufficiently large,*

$$\kappa_i > \frac{L_i^2}{m_i^2} - \rho\underline{\lambda}(\boldsymbol{D} + \boldsymbol{A}) > 0, \quad \forall i \in \mathcal{V}, \tag{11}$$

where $L_i > 0$ *and* $m_i > 0$ *are positive parameters of Lipschitz continuity and strong convexity respectively. Then, the nested optimization for the hyperparameter update* (10b) *admits a closed-form solution, resulting in the iterative optimization scheme of DEC-apx-GP,*

$$\boldsymbol{p}_i^{(s+1)} = \boldsymbol{p}_i^{(s)} + \rho \sum_{j \in \mathcal{N}_i}\left(\boldsymbol{\theta}_i^{(s)} - \boldsymbol{\theta}_j^{(s)}\right), \tag{12a}$$

$$\begin{aligned}\boldsymbol{\theta}_i^{(s+1)} = \frac{1}{\kappa_i + 2\text{card}(\mathcal{N}_i)\rho}\Bigg(&\rho \sum_{j \in \mathcal{N}_i} \boldsymbol{\theta}_j^{(s)} - \nabla \mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) + \\ &\left(\kappa_i + \text{card}(\mathcal{N}_i)\rho\right)\boldsymbol{\theta}_i^{(s)} - \boldsymbol{p}_i^{(s+1)}\Bigg), \end{aligned} \tag{12b}$$

that converges to a stationary solution $(\boldsymbol{\theta}_i^\star, \boldsymbol{p}^\star)$ *of (P3) in* (7) *for all agents* $i \in \mathcal{V}$.

Proof: (Sketch) After rearrangement and manipulation, the local objective $\mathcal{Q}_i$ *of the nested optimization* (10b) *is expressed as,*

$$\begin{aligned}\mathcal{Q}_i(\boldsymbol{\theta}_i) = \boldsymbol{\theta}_i^\intercal \Big(&\nabla \mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) - \left(\kappa_i + \text{card}(\mathcal{N}_i)\rho\right)\boldsymbol{\theta}_i^{(s)} \\ &+ \boldsymbol{p}_i^{(s+1)} - \rho \sum_{j \in \mathcal{N}_i} \boldsymbol{\theta}_j^{(s)}\Big) + \left(\frac{\kappa_i}{2} + \text{card}(\mathcal{N}_i)\rho\right)\boldsymbol{\theta}_i^\intercal \boldsymbol{\theta}_i. \end{aligned} \tag{13}$$

Next, we show that the Hessian of (13) *is positive definite* $\mathcal{H}_{\mathcal{Q}_i} = \nabla^2_{\boldsymbol{\theta}_i}\mathcal{Q}_i \succ 0$ *that leads to convex and quadratic local objective* $\mathcal{Q}_i$*. Thus, a closed-form solution of the nested optimization is derived by computing* $\partial\mathcal{Q}_i/\partial\boldsymbol{\theta}_i = \mathbf{0}$,

$$\begin{aligned}\boldsymbol{\theta}_i = \frac{1}{\kappa_i + 2\text{card}(\mathcal{N}_i)\rho}\Bigg(&\rho \sum_{j \in \mathcal{N}_i} \boldsymbol{\theta}_j^{(s)} - \nabla \mathcal{L}_i\left(\boldsymbol{\theta}_i^{(s)}\right) + \\ &\left(\kappa_i + \text{card}(\mathcal{N}_i)\rho\right)\boldsymbol{\theta}_i^{(s)} - \boldsymbol{p}_i^{(s+1)}\Bigg). \end{aligned}$$

The rest of the proof is a consequence [44, Theorem 1]. ■

Remark 2. *The condition to select the penalty parameter* κ_i (11) *depends on the graph topology as the minimum eigenvalue of the degree and adjacency matrix is required* $\underline{\lambda}(\boldsymbol{D} + \boldsymbol{A})$*. Thus, the stronger the network connectivity, the faster the convergence of the proposed DEC-apx-GP* (12).

The DEC-apx-GP routine is provided in Algorithm 1. 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 $\boldsymbol{\theta}_j^{(s)}$ from its neighborhood, then the sum of

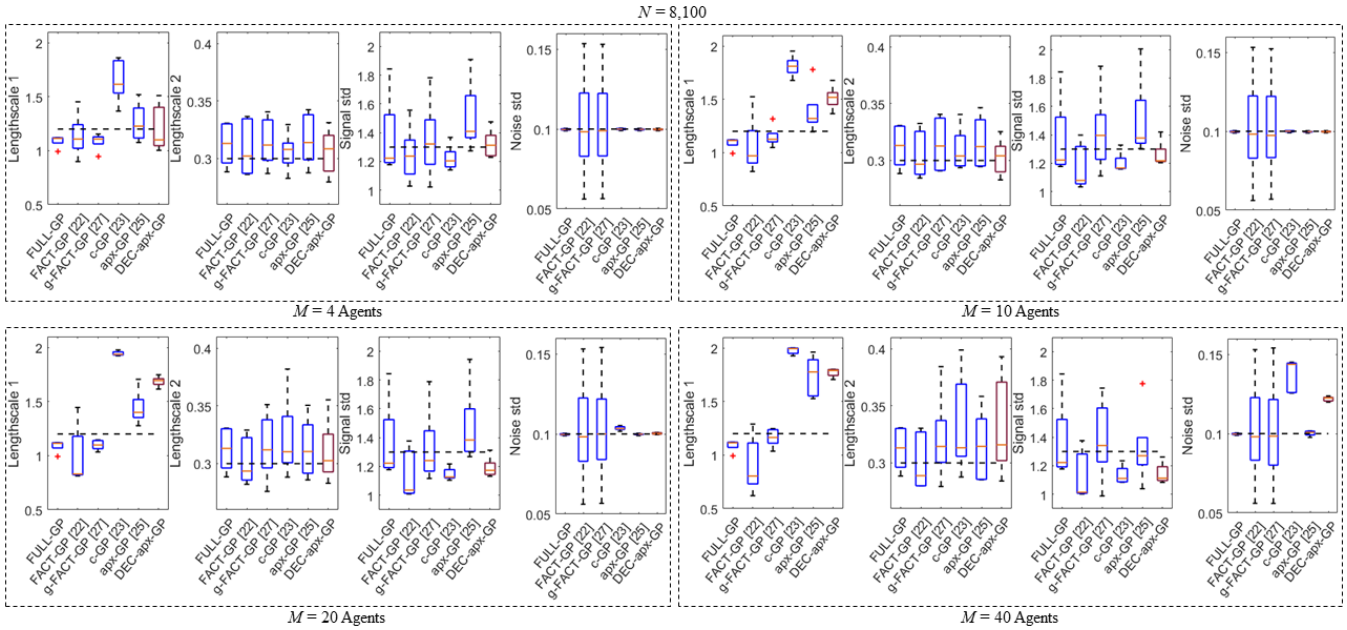

Fig. 3. Accuracy of GP hyperparameter training using $N = 8,100$ on five generative GP functions data for four fleet sizes and 50 replications. The true values are demonstrated with a black dotted line. The existing GP training methods are shown in blue boxes and the proposed in maroon coloured box.

Algorithm 1 DEC-apx-GP

Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $k(\cdot,\cdot)$, ρ, $\mathcal{N}_i$, α, $s^{\text{end}}_{\text{DEC-apx-GP}}$
Output: $\hat{\boldsymbol{\theta}}$

1: **for** $s = 1$ to $s^{\text{end}}_{\text{DEC-apx-GP}}$ **do**
2: **for each** $i \in \mathcal{V}$ **do**
3: communicate $\boldsymbol{\theta}_i^{(s)}$ to neighbors $\mathcal{N}_i$
4: $\boldsymbol{p}_i^{(s+1)} \leftarrow \texttt{duals}(\boldsymbol{p}_i^{(s)}, \boldsymbol{\theta}_i^{(s)}, \{\boldsymbol{\theta}_j^{(s)}\}_{j\in\mathcal{N}_i}, \rho)$ (12a)
5: $\boldsymbol{\theta}_i^{(s+1)} \leftarrow \texttt{prim}(\boldsymbol{p}_i^{(s+1)}, \boldsymbol{\theta}_i^{(s)}, \{\boldsymbol{\theta}_j^{(s)}\}_{j\in\mathcal{N}_i}, \rho, \alpha, \mathcal{D}_i)$ (12b)
6: **end for**
7: **end for**

the dual variables vector is updated (12a) to obtain $\boldsymbol{p}_i^{(s+1)}$. Next, every agent i computes analytically the hyperparameters $\boldsymbol{\theta}_i^{(s+1)}$ (12b). The method iterates until a predefined iteration number $s^{\text{end}}_{\text{DEC-apx-GP}}$ is reached. The proposed method is decentralized, requiring exclusively neighbor-wise communication as shown in Fig. 2. Note that information exchange does not involve any data exchange (Assumption 2) to satisfy the requirements of federated learning.

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 $\boldsymbol{C}_{\theta,i}^{-1}$ just once at every ADMM iteration (12). Every agent i occupies $\mathcal{O}(N^2/M^2 + D(N/M))$ memory to store $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{D}_i$, $\boldsymbol{p}_i^{(s)}$, $\boldsymbol{\theta}_i^{(s)}$, and $\{\boldsymbol{\theta}_j^{(s)}\}_{j\in\mathcal{N}_i}$. The total number of communications for each agent is $\mathcal{O}(s^{\text{end}}_{\text{DEC-apx-GP}}(D+2))$ to transmit the hyperparameter vector for all iterations.

IV. Numerical Experiments

In this section, we perform numerical experiments with synthetic data of known hyperparameters values 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 using [45] on an Intel Core i7-6700 CPU @3.40 GHz with 32.0 GB memory RAM.

We conduct 2,000 numerical experiments where we generate datasets by using the observation model (1) and the separable squared exponential kernel (2) with hyperparameters $\boldsymbol{\theta} = (l_1, l_2, \sigma_f, \sigma_\epsilon)^\top = (1.2, 0.3, 1.3, 0.1)^\top$. 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. Moreover, we equally partition the space of interest $\mathsf{S} = [0,2]^2$ along the x-axis according to fleet sizes $M = \{4, 10, 20, 40\}$, and assign local datasets. We compare the global GP training FULL-GP; the centralized methods FACT-GP [22], generalized FACT-GP (g-FACT-GP) [27], c-GP [23], and apx-GP [25]; and the proposed decentralized DEC-apx-GP. The decentralized network follows a path graph topology that is the most parsimonious connected network. Thus, we study the worst case scenario in terms of network connectivity (Remark 2). The penalty parameter of the augmented Lagrangian is set to $\rho = 500$, the decentralized ADMM tolerance for convergence $\text{TOL}_{\text{ADMM}} = 10^{-3}$, and the regulation of the approximation $\kappa_i = 5,000$ for all $i \in \mathcal{V}$. After $s^{\text{end}}_{\text{DEC-apx-GP}} = 100$ communication rounds the proposed DEC-apx-GP is completed.

In Fig. 3, we show the boxplots of the estimated hyperparameters using $N = 8,100$ data. Blue boxes illustrate existing GP training methods and the maroon box represents the proposed method. The corresponding average computation time per agent and the communication rounds are shown in Table I. For the case of $M = 4$ agents, all methods provide accurate hyperparameters estimates except of the c-GP on l_1. In terms of computation time, c-GP is the more demanding

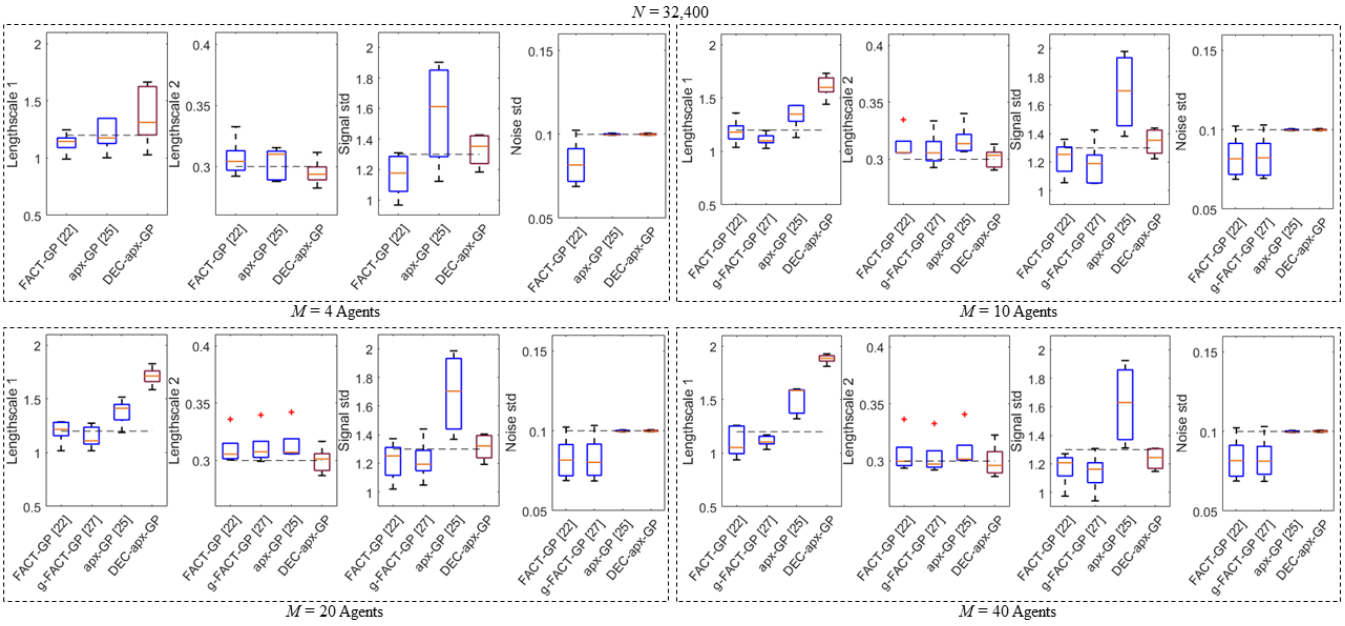

Fig. 4. Accuracy of GP hyperparameter training using $N = 32,400$ data on five generative GP functions for four fleet sizes and 50 replications. The true values are demonstrated with a black dotted line. The existing GP training methods are shown in blue boxes and the proposed in maroon box.

TABLE I
TIME & COMMUNICATION ROUNDS OF GP TRAINING METHODS

M	Method	$N = 8,100$ Time [s]	$N = 8,100$ Comms s^{end}	$N = 32,400$ Time [s]	$N = 32,400$ Comms s^{end}
	FULL-GP	2,114.2	-	>3,000	-
4	FACT-GP [22]	75.9	186.0	2,361.9	196.0
	g-FACT-GP [27]	332.1	160.0	>3,000	-
	c-GP [23]	404.1	141.4	-	-
	apx-GP [25]	26.8	43.6	817.6	45.2
	DEC-apx-GP	61.9	100	1,821.3	100
10	FACT-GP [22]	9.8	179.6	228.2	194.2
	g-FACT-GP [27]	31.8	131.8	1,035.6	155.2
	c-GP [23]	92.1	193.8	-	-
	apx-GP [25]	3.8	47.8	88.8	46.8
	DEC-apx-GP	8.4	100	188.8	100
20	FACT-GP [22]	2.6	172.6	46.6	226.2
	g-FACT-GP [27]	7.0	127.2	199.4	167.6
	c-GP [23]	31.4	127.8	-	-
	apx-GP [25]	1.3	56.2	18.3	49.8
	DEC-apx-GP	2.2	100	36.9	100
40	FACT-GP [22]	0.5	139.6	9.1	160.0
	g-FACT-GP [27]	1.8	112.2	30.9	128.6
	c-GP [23]	8.9	66.6	-	-
	apx-GP [25]	0.3	56.4	4.6	54.4
	DEC-apx-GP	0.5	100	8.2	100

method, while FACT-GP, apx-GP, and the proposed DEC-apx-GP converge very fast, outperforming FULL-GP by two orders of magnitude for similar model estimation accuracy. As we increase the fleet size ($M = 10$, $M = 20$, and $M = 40$ agents), the hyperparameter estimation accuracy deteriorates for all centralized and decentralized methods except g-FACT-GP, but the local computations are becoming significantly low (Table I). In particular, DEC-apx-GP produces reasonable estimates for all hyperparameters other than l_1 and requires three orders of magnitude lower computations for each robot than FULL-GP.

We present the boxplots of the estimated hyperparameters using $N = 32,400$ data in Fig. 4. The FULL-GP and c-GP cannot be implemented for $N = 32,400$ data, as they entail significantly high computations (Remark 1). For $M = 4$ agents, g-FACT-GP exceeds the time limit of 3,000 s for convergence. All methods are computationally expensive as each agent i is assigned $N_i = 32,400/4 = 8,100$ data, yet apx-GP is the fastest. In addition, all feasible methods produce accurate hyperparameter estimates. As we increase the number of agents ($M = 10$, $M = 20$, and $M = 40$ agents), the number of data is distributed to local agents, and thus g-FACT-GP can be implemented. Since the number of data is high, all centralized methods produce accurate hyperparameters estimates. The proposed decentralized method produces accurate hyperparameter estimates other than l_1 and requires competitive computations for its implementation.

V. CONCLUSION AND FUTURE WORK

This paper proposes a decentralized federated learning method for GP training in teams of robots. The proposed decentralized method is suitable for long-term multi-robot missions where centralized network topologies cannot be implemented and inter-agent sharing of local data is prohibited. The closed-form solution of the nested optimization of decentralized ADMM improves the GP model accuracy and scalability. DEC-apx-GP is shown to achieve competitive accuracy in hyperparameter estimates for small and medium fleet sizes. Ongoing work focuses on improving the hyperparameter estimation accuracy of GP training for large fleets.

REFERENCES

[1] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2006.

[2] 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.

[3] 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.

[4] 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.

[5] S. Manjanna, A. Q. Li, R. N. Smith, I. Rekleitis, and G. Dudek, "Heterogeneous multi-robot system for exploration and strategic water sampling," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 4873–4880.

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

[7] M. Ghaffari Jadidi, J. Valls Miro, and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Autonomous Robots*, vol. 42, pp. 273–290, 2018.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] D. Jang, J. Yoo, C. Y. Son, D. Kim, and H. J. Kim, "Multi-robot active sensing and environmental model learning with distributed Gaussian process," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5905–5912, 2020.

[14] 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.

[15] 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.

[16] W. Chen, R. Khardon, and L. Liu, "Adaptive robotic information gathering via non-stationary Gaussian processes," *International Journal of Robotics Research*, 2023.

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

[18] 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.

[19] 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.

[20] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

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

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

[23] 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.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, 2011, vol. 3, no. 1.

[25] 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.

[26] 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.

[27] 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.

[28] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A survey of distributed optimization methods for multi-robot systems," *arXiv preprint arXiv:2103.12840*, 2021.

[29] 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. 15 369–15 378, 2021.

[30] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian process regression: Active data selection and test point rejection," in *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 3, 2000, pp. 241–246.

[31] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies." *Journ. of Machine Learning Research*, vol. 9, no. 2, 2008.

[32] R. B. Gramacy and H. K. Lee, "Adaptive design and analysis of supercomputer experiments," *Technometrics*, vol. 51, no. 2, pp. 130–145, 2009.

[33] R. B. Gramacy, *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Boca Raton, Florida: Chapman Hall/CRC, 2020.

[34] G. P. Kontoudis and M. Otte, "Closed-form active learning using expected variance reduction of Gaussian process surrogates," in *American Control Conference*, 2023, pp. 4626–4632.

[35] C. N. Mavridis, G. P. Kontoudis, and J. S. Baras, "Sparse Gaussian process regression using progressively growing learning representations," in *IEEE Conference on Decision and Control*, 2022, pp. 1454–1459.

[36] G. P. Kontoudis and M. Otte, "Adaptive exploration-exploitation active learning of Gaussian processes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[37] 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.

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

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (GPML) toolbox," *Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, 2010.