Received 29 January, 2025; revised 12 September, 2025; accepted 02 December, 2025; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/TMLCN.2022.1234567

Multi-Agent Federated Learning using Covariance-Based Nearest Neighbor Gaussian Processes

George P. Kontoudis[1], Member, IEEE and Daniel J. Stilwell[2], Member, IEEE

[1]Department of Mechanical Engineering, Colorado School of Mines, Golden, CO 80401, USA
[2]Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA

Corresponding author: George P. Kontoudis (email: 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 In this paper, we propose scalable methods for Gaussian process (GP) prediction in decentralized multi-agent systems. Multiple aggregation techniques for GP prediction are decentralized with the use of iterative and consensus methods. Moreover, we introduce a covariance-based nearest neighbor selection strategy that leverages cross-covariance similarity, enabling subsets of agents to make accurate predictions. The proposed decentralized schemes preserve the consistency properties of their centralized counterparts, while adhering to federated learning principles by restricting raw data exchange between agents. We validate the efficacy of the proposed decentralized algorithms with numerical experiments on real-world sea surface temperature and ground elevation map datasets across multiple fleet sizes.

INDEX TERMS Federated learning, Gaussian processes, multi-agent systems, nearest neighbor methods, prediction algorithms.

I. INTRODUCTION

MULTI-AGENT learning has received significant attention in recent years, driven by the capability to simultaneously gather observations from remote locations. In addition, local agents enhance resource efficiency, as they are equipped with limited computation and communication resources. Remote data acquisition facilitates the formation of large datasets that enable reliable and robust learning of latent fields. Thus, teams of agents are favored for learning due to their ability to rapidly generate large datasets. In typical multi agent-systems, central clusters collect large datasets from local nodes and process the information. However, centralizing the collection of data involves the transmission of sensitive information from all agents, which conflicts with the EU/UK general data protection regulation (GDPR) [1]. To overcome information sharing limitations, federated learning (FL) methodologies enable multiple entities to collaboratively learn a shared model of a latent field with no data exchange [2]. Although FL methods are primarily designed for implementation in centralized networks, autonomous systems and edge devices favor decentralized communication networks due to the limited computation and communication capabilities of local nodes [3].

Our objective in this work is to formulate decentralized methodologies for Gaussian process (GP) prediction with reduced computations and limited inter-agent information exchange. We develop several decentralized approximate methods to perform GP prediction with aggregation of GP experts [4], using iterative methods and consensus protocols [5]–[7]. In addition, we introduce a covariance-based nearest neighbor selection technique that identifies statistically correlated agents to participate in GP prediction. The proposed methods are illustrated in Figure 1. An exploration task is assigned to a team of agents operating in a decentralized communication network. Each agent collects data from the unknown field and maintains a local dataset. The proposed decentralized GP prediction method (middle row in Figure 1) involves three steps: i) making predictions based on local datasets; ii) communication among all agents to aggregate local predictions; and iii) reaching consensus among all agents on a global prediction. The covariance-based nearest neighbor GP prediction method (bottom row in Figure 1) comprises of five steps: i) making predictions based on local

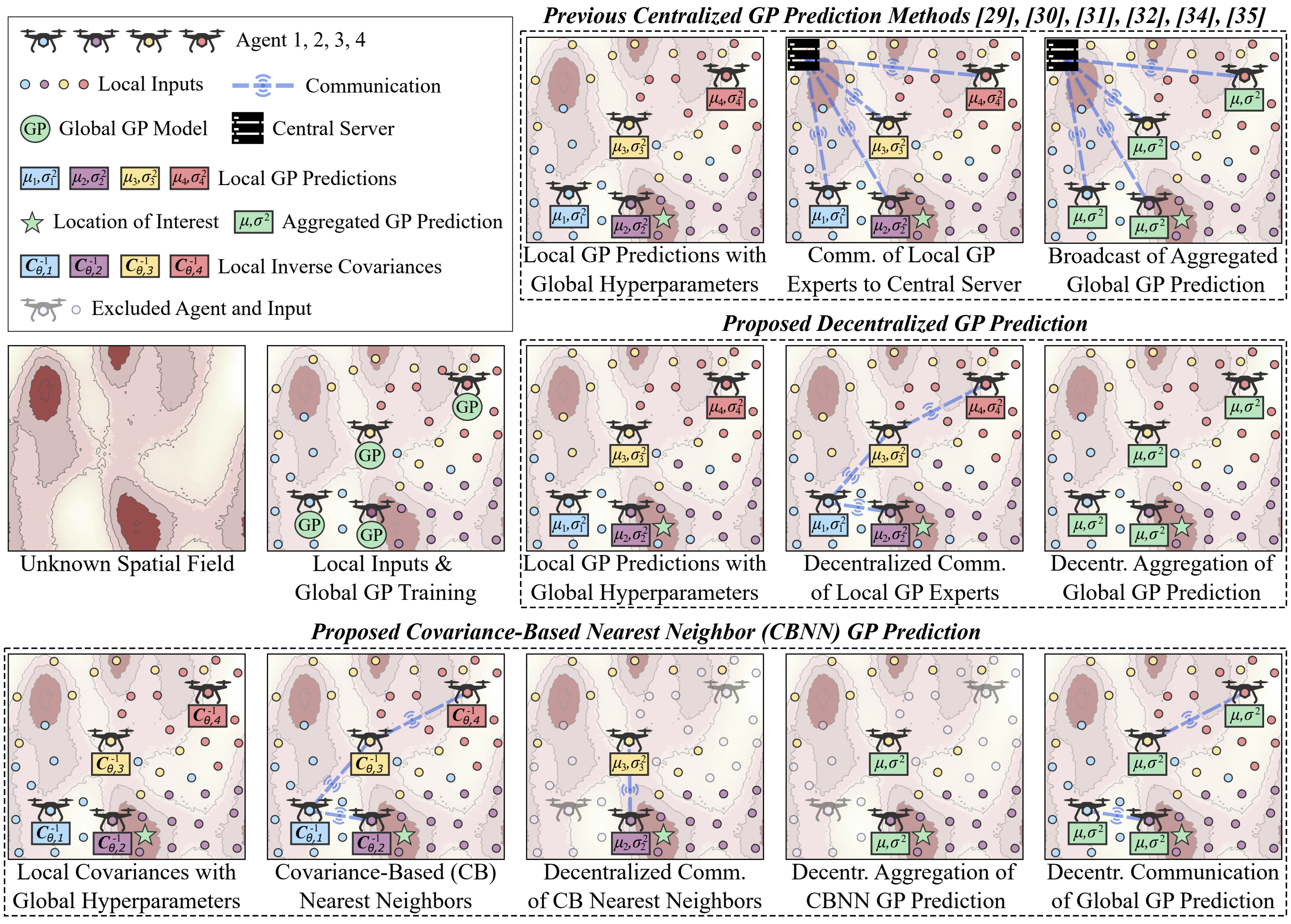
Agent 1, 2, 3, 4
Local Inputs
Communication
Global GP Model
Central Server
Local GP Predictions
Location of Interest
Aggregated GP Prediction
Local Inverse Covariances
Excluded Agent and Input
Previous Centralized GP Prediction Methods [29], [30], [31], [32], [34], [35]
Local GP Predictions with Global Hyperparameters
Comm. of Local GP Experts to Central Server
Broadcast of Aggregated Global GP Prediction
Proposed Decentralized GP Prediction
Unknown Spatial Field
Local Inputs & Global GP Training
Local GP Predictions with Global Hyperparameters
Decentralized Comm. of Local GP Experts
Decentr. Aggregation of Global GP Prediction
Proposed Covariance-Based Nearest Neighbor (CBNN) GP Prediction
Local Covariances with Global Hyperparameters
Covariance-Based (CB) Nearest Neighbors
Decentralized Comm. of CB Nearest Neighbors
Decentr. Aggregation of CBNN GP Prediction
Decentr. Communication of Global GP Prediction

FIGURE 1. **A covariance-based nearest neighbor GP prediction method in decentralized teams using federated learning. While we demonstrate an unknown spatial field for exploration, our methodologies are adaptable to higher input dimensions $D > 2$. Local datasets are depicted as colored circles, each corresponding to the color of the respective agent. Federated learning is facilitated with the constraint on data communication between agents.**

datasets; ii) communication among all agents to identify the covariance-based nearest neighbors; iii) communication among covariance-based nearest neighbors to aggregate local predictions; iv) reaching consensus among covariance-based nearest neighbors on a global prediction; v) broadcasting of global prediction to excluded agents.

Gaussian Processes [8], [9] have shown notable success in learning latent fields for numerous applications including environmental monitoring [10], search and rescue [11], rehabilitation [12], and multi-agent missions [13]–[25]. The main disadvantage of GPs is the poor scalability with large datasets. In particular, GPs entail $\mathcal{O}(N^3)$ computations for training and $\mathcal{O}(N^2)$ computations for prediction, where N represents the dataset size. For multi-robot systems, GP implementation becomes infeasible due to limited computational resources available across multiple robots. In addition, even if all data are centralized, the slow execution of GPs hinders real-time decision making in field robots. Besides computational limitations in multi-agent systems, GPs require high communication overhead for both centralized and decentralized network topologies [3]. This may lead to network congestion, information loss, and security issues. Consequently, multiple approximation methods have been discussed in the literature for scalable execution of GPs that are surveyed in [4]. These methods consists of two major research directions for GP approximations based on global and local techniques.

Global approximation techniques employ sparse datasets N_{sub} either by introducing pseudo-inputs or by selecting a subset of the full dataset [26]–[28]. The selected subset is considered to be significantly smaller than the full dataset $N_{\text{sub}} \ll N$ and thus leads to lower computations. Sparse GPs are attractive for modeling spatial fields with mobile sensor networks [29]. Chen *et al.* [14] employed sparse methods for traffic prediction. The authors in [13] truncated the dataset to perform spatiotemporal modeling with GPs. Global methods require the communication of the full dataset which increases significantly the inter-agent communications. Moreover, the interpolation property is violated with pseudo-inputs as they are not true observations.

The local GP approximation methods are centralized approaches with a server-client structure. The main idea is to aggregate local sub-models produced by local subsets of the observations [30]–[33]. In other words, every sub-model makes a local prediction, and then the central node aggregates to a single prediction. It is proved in [34, Proposition 1] that the local methods [30]–[32] are *inconsistent*, i.e. as the observation size grows to infinity, the aggregated predictions do not converge to the true values. Although [33] is consistent in terms of prediction mean, it produces overconfident prediction variances [35, Proposition 1]. Subsequently, the authors in [36] proposed the nested pointwise aggregation of experts (NPAE) that takes into account the covariance between sub-models and produces consistent predictions. The price to achieve consistency in NPAE comes with much higher computational complexity in the central node. Liu *et al.* [35] introduced a computationally efficient and consistent methodology, termed as generalized robust Bayesian committee machine (grBCM). The latter entails additional communication between agents to enrich local datasets with a global random dataset. In addition, both NPAE and grBCM are centralized techniques—not well-suited for multi-agent systems [37].

Gaussian process training is crucial for ensuring the validity of the GP model used in GP prediction [38, Chapter 3]. Distributed GP training often relies on the assumption of independence, where each local dataset is considered statistically independent. Most existing methods are focused on centralized networks [32], [35]. Since the maximum likelihood estimation problem for multi-agent systems yields a distributed optimization problem, the Alternating Direction Method of Multipliers (ADMM) [39] is commonly used in the literature[40], [41]. Recently, we extended distributed GP training to decentralized networks, applicable for both small and medium fleet sizes [24] as well as large networks [42]. However, the focus of this paper is on distributed GP prediction for decentralized multi-agent systems that follows the estimation of the hyperparameters through GP training.

A decentralized method for the computation of spatio-temporal GP predictions is proposed in [43]. In [44], a decentralized technique for spatial GP prediction with localization uncertainty is presented. Both [43] and [44] employ the Jacobi over-relaxation (JOR), which requires a complete graph topology, i.e. every node must communicate to every other node. That is a conservative network topology and is not common in multi-agent systems [37]. For non-complete topologies, JOR entails flooding before every iteration. In flooding, each agent broadcasts all input packets to its neighbors [45]. Thus, the communication requirements of JOR are high. Pillonetto *et al.* [16] proposed sub-optimal methods to distributively predict a latent function by employing orthonormal eigenfunctions, computed by the Karhunen-Loève expansion of a GP kernel. An extension of this work to multi-robot systems with online information gathering is discussed in [46]. This is a promising line of research for GPs in decentralized networks, but our focus is on decentralized and scalable GP prediction with aggregation methods. Nevertheless, computing orthonormal eigenfunctions in closed-form is not feasible for all kernels and may yield significant storage requirements.

Recently, nearest neighbor methods for GP prediction have attracted attention in the literature [47]–[52]. In [53], the authors proposed a methodology that combines nearest neighbors and local approximation for GP prediction. However, arbitrary selection of nearest neighbors may lead to poor approximations [48] and suffers from prediction discontinuities [36]. In [48], [49], the authors propose a nearest neighbor Gaussian process (NNGP) approximation by employing sparse precision matrices for spatiotemporal fields. An extension of NNGP with stochastic variational inference is introduced in [50], where inducing inputs are utilized to perform a low-rank approximation of the covariance matrix. In [51], the authors apply the normal inverse Wishart distribution as a prior for non-isotropic nearest neighbor Gaussian process prediction. However, these methods typically require a central entity or a complete graph topology, which is often impractical for multi-agent systems [3]. In contrast, our work focuses on Gaussian processes for decentralized teams, where nearest neighbors are selected based on a cross-covariance similarity metric.

A module-driven GP prediction method is discussed in [54], where sparse GP methods are employed within local modules. This approach yields independent posterior distributions through variational inference by using large-dimensional integral operators. However, variational inference performs poorly for non-smooth latent fields and requires a high number of pseudo-inputs [55]. Patchwork Kriging (PK) [56] is a consistent GP predeiction method that requires central knowledge of the entire dataset to partition the global dataset to independent local regions. Local GPs are trained independently in each region and PK aggregates the local GP models by enforcing continuity across region boundaries. Our focus is on federated learning with no central access to the entire global dataset. In addition, PK solves the inverse problem of decentralized multi-agent systems where local datasets are assigned to agents after partitioning the area. In [57], the authors provide convergence bounds for a federated GP method that uses an aggregation technique and stochastic gradient descent. Unlike our approach, these federated GP methods depend on a centralized network topology for aggregation. Our focus is on federated GP prediction for decentralized multi-agent systems.

To facilitate a high-level comparison, we evaluate methods across five key criteria in Table 1 consisting of: (1) Distribution type (centralized, decentralized, or non-distributed), capturing whether computation and data remain local or rely on a central server; (2) Network topology requirements (connected versus complete), which specify the communication structure necessary for prediction; (3) Prediction consistency, assessing whether a method produces asymptotically consis-

TABLE 1. Time, Space, and Communication Complexity for Centralized GP Aggregated Prediction

Method	Distribution	Network	Consistency	Communication	Federated
PoE [31]	Centralized	Connected	Inconsistent	Whole	Non-federated
gPoE [33]	Centralized	Connected	Inconsistent	Whole	Non-federated
BCM [30]	Centralized	Connected	Inconsistent	Whole	Non-federated
rBCM [32]	Centralized	Connected	Inconsistent	Whole	Non-federated
grBCM [35]	Centralized	Connected	Consistent	Whole	Non-federated
NPAE [36]	Centralized	Connected	Consistent	Whole	Non-federated
SVGP [28]	Non-distributed	Single agent	Inconsistent	N/A	N/A
NNGP [48]	Centralized	Connected	Consistent	NN	Non-federated
PK [56]	Centralized	Connected	Consistent	Whole	Non-federated
DEC-(NN)-NPAE [58]	Decentralized	Complete – DEC-NPAE Connected – DEC-NN-NPAE	Consistent	(NN)-variant	Federated
DEC-NPAE⋆	Decentralized	Complete	Consistent	Whole	Federated
DEC-(NN)-(g)PoE	Decentralized	Connected	Inconsistent	(NN)-variants	Federated
DEC-(NN)-(gr)(r)BCM	Decentralized	Connected	Consistent – DEC-(NN)-grBCM	(NN)-variants	Federated – DEC-(NN)-(r)BCM

DEC-(NN)-(g)PoE refers to: DEC-PoE, DEC-gPoE, DEC-NN-PoE, and DEC-NN-gPoE. Similarly, DEC-(NN)-(gr)(r)BCM denotes six methods.

tent estimates; (4) Communication pattern (nearest-neighbor versus whole-network), reflecting the scalability of information exchange; and (5) Federation capability, indicating whether methods can operate in a federated setting without raw data aggregation. The proposed algorithms (brown font) and our previous work [58] are the only methods that can be employed in decentralized multi-agent systems and promote federated learning.

The contribution of this paper is threefold. First, we introduce a covariance-based nearest neighbor (CBNN) technique that selects statistically correlated agents based on a cross-covariance similarity metric for GP prediction and provide a consistency proof. The CBNN is applicable to the decentralized versions of PoE, gPoE, BCM, rBCM, and grBCM. Moreover, CBNN allows the use of a distributed algorithm for solving systems of linear equations (DALE) [7], [59] which replaces JOR in the decentralized NPAE and relaxes the graph topology requirements from complete to connected. Second, we decentralize the implementation of several aggregation of GP experts methods including PoE [31], gPoE [33], BCM [30], rBCM [32], and grBCM [35], for connected graph topologies. Lastly, we decentralize the implementation of NPAE [36] for complete graph topologies, by combining Jacobi over-relaxation (JOR) [5, Chapter 2.4] with DAC. In addition, we introduce a technique to recover the optimal relaxation factor of JOR [60] for complete graph topologies by using the power method (PM) [61, Chapter 8] ensuring faster convergence.

Two of the thirteen methods discussed in this work (DEC-NPAE and DEC-NN-NPAE) were originally introduced in our prior work [58], but we report them here for completeness. In this paper, we significantly extend and improve upon those initial methods. The newly proposed approaches consistently outperform DEC-NPAE and DEC-NN-NPAE in prediction accuracy, uncertainty quantification, as well as communication and computation efficiency. The distinction between the earlier DEC-NPAE and the new DEC-NPAE⋆ is illustrated in Figure 3, where we now distributively estimate the optimal relaxation factor; their comparative performance is presented in Figure 10. For DEC-NN-NPAE, Table 6 qualitatively demonstrates that all newly proposed methods achieve superior results across accuracy, uncertainty quantification, and efficiency. In addition, we provide the complete proof of Lemma 6 (cf. [58, Proposition 3]) and a detailed complexity analysis covering both existing centralized methods and all proposed decentralized methods for GP prediction. These methods are also fully documented in the corresponding PhD dissertation [62] and arXiv preprint [63], which serve as technical reports including both GP training [42] and prediction.

The remainder of this paper is organized as follows. In Section II we discuss centralized GP prediction methods and formulate the problem. Section IV introduces the decentralized GP prediction methods and Section V the covariance-based nearest neighbors. Section VI provides numerical experiments and results, Section VII discusses the underlying assumptions, and Section VIII concludes the paper.

II. Preliminaries and Problem Statement

In this section, we introduce algebraic graph theory, discuss GP regression, overview existing centralized methods, and state the problem of decentralized GP prediction.

A. Foundations

The notation here is standard. We denote the set of all non-negative real numbers $\mathbb{R}_{\geq 0}$ and the identity matrix of dimension $n \times n$ by I_n. The s-th iteration of an update law is denoted by a superscript in parenthesis $y^{(s)}$. We denote the cardinality of the set K by $\text{card}(K)$, the L_2 norm by $\|\cdot\|_2$, and the infinity norm by $\|\cdot\|_\infty$. The maximum and minimum eigenvalue of matrix $\boldsymbol{A}$ is denoted $\overline{\lambda}(\boldsymbol{A})$ and $\underline{\lambda}(\boldsymbol{A})$ respectively. A vector $\boldsymbol{x} \in \mathbb{R}^N$ can be expressed as $\{x_i\}_{i=1}^N$. The communication complexity $\mathcal{O}(\cdot)$ represents the total number of transmitted bits throughout an algorithm [37, Chapter 3]. Time and space complexity are denoted $\mathcal{O}(\cdot)$, and indicate the maximum number of computations required and maximum space needed at any point during the algorithm respectively. The complexities are evaluated based on the the total number of observations N, the input dimension D, and the size of the team M. Moreover, s^{end} represents the maximum number of iterations needed for an algorithm to reach convergence.

Let a team consist of M agents, each capable of performing local computations. The network is represented by a set of nodes $\mathcal{V} = 1, \dots, M$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ in an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The nodes correspond to agents and the edges represent the communication links between them. In an undirected graph, edges between nodes are bidirectional. The neighborhood set of the i-th node is described by $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$. The adjacency matrix is denoted $\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{M \times M}$, where $a_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. The degree matrix is denoted $\boldsymbol{D} = [d_{ij}] \in \mathbb{R}^{M \times M}$ with $d_{ii} = \sum_{j=1}^M a_{ij}$. Let us define the graph Laplacian as $\mathcal{L} := \boldsymbol{D} - \boldsymbol{A}$. The maximum number of neighbors in the graph $\mathcal{G}$ is called maximum degree $\Delta = \max_i\{\sum_{j \neq i} a_{ij}\}$. We employ the Perron matrix $\mathcal{P} := I_M - \epsilon\mathcal{L}$, where $\epsilon \in (0, 1]$. The diameter of a graph $\mathcal{G}$ is denoted $\text{diam}(\mathcal{G})$ and represents the shortest distance between any nodes.

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

Graph connectivity can significantly affect the behavior of algorithms, ranging from a *one-hop line graph* (most parsimonious topology) to a *complete graph* (most connected topology). In a one-hop line graph, every agent i can communicate only with its one-hop neighbors, i.e., $\Delta = 2$, and any link failure results in network disconnection. In contrast, in a complete graph, every agent i is connected with every other agent $j \neq i$, i.e., $\Delta = M - 1$, and a link failure does not necessarily compromise connectivity. To capture intermediate scenarios, we also consider a two-hop line graph, where each agent communicates with its two nearest neighbors on either side, i.e., $\Delta = 4$, and the Erdős-Rényi random graph, where edges are formed probabilistically with parameter p, providing a flexible model of sparse but connected networks. In this paper, the line graph is used as the most parsimonious connected topology, serving as a lower bound for the performance of the proposed algorithms, while the two-hop and Erdős-Rényi graphs demonstrate improved communication efficiency and execution time under stronger connectivity.

B. Gaussian Processes

Let a model observation that follows,

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

where $\boldsymbol{x} \in \mathbb{R}^D$ represents the input location with D being the dimension of the input space, $f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$ follows a zero-mean GP with covariance function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$. The term $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ represents independent and identically distributed (i.i.d.) measurement noise with variance $\sigma_\epsilon^2 > 0$. To formulate the covariance matrix we use the separable squared exponential covariance function,

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

where $l_d > 0$ is the length-scale hyperparameter at the corresponding d-th direction to the input space and $\sigma_f^2 > 0$ is the signal variance. The objective of GPs is to infer an unknown function f based on the given data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$, where $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ represents the input points and $\boldsymbol{y} = \{y_n\}_{n=1}^N$ the corresponding outputs, with N being the total number of observations.

1) Training

The goal of GP training is to estimate hyperparameters of the covariance function $\boldsymbol{\theta} = (l_1, \dots, l_D, \sigma_f, \sigma_\epsilon)^\intercal \in \Theta \subset \mathbb{R}^{D+2}$. The log-likelihood takes the form of,

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

where $\boldsymbol{C}_\theta = \boldsymbol{K} + \sigma_\epsilon^2 I_N \succ 0 \in \mathbb{R}^{N \times N}$ is the covariance and $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X}) \succeq 0 \in \mathbb{R}^{N \times N}$ the correlation matrix.

2) Prediction

After estimating the hyperparameters $\hat{\boldsymbol{\theta}}$, the predictive distribution of the unknown location $\boldsymbol{x}_* \in \mathbb{R}^D$ conditioned on the data, $p(y_* \mid \mathcal{D}, \boldsymbol{x}_*) \sim \mathcal{N}(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$ is characterized by the prediction mean and variance,

$$\mu_{\text{full}}(\boldsymbol{x}_*) = \boldsymbol{k}_*^\intercal \boldsymbol{C}_\theta^{-1}\boldsymbol{y}, \tag{3}$$

$$\sigma^2_{\text{full}}(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_*^\intercal \boldsymbol{C}_\theta^{-1}\boldsymbol{k}_*, \tag{4}$$

where $k_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*) \in \mathbb{R}$ and $\boldsymbol{k}_* = k(\boldsymbol{X}, \boldsymbol{x}_*) \in \mathbb{R}^N$.

3) Complexity

After estimating the hyperparameters in the GP training, we store the covariance inverse $\boldsymbol{C}_\theta^{-1}$ as well as the dataset $\mathcal{D}$, which requires $\mathcal{O}(N^2 + DN)$ space. For agents with limited memory RAM capacity, space complexity is more restrictive than time complexity. The prediction mean (3) and variance (4) entail $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ computations respectively.

C. Problem Formulation

Consider a team of M agents, each capable of performing local computations. Each agent i collects observations and maintains its own local dataset $\{\mathcal{D}_i = \{\boldsymbol{X}_i, \boldsymbol{y}_i\}\}_{i=1}^M$. The global dataset is the union of all local datasets $\mathcal{D} = \cup_{i=1}^M \mathcal{D}_i$. To enable federated learning, the communication of local datasets $\mathcal{D}_i$ between agents is restricted. In practice, even if all agents had access to the global dataset $\mathcal{D}$, the computational complexity of GPs (Section 3) becomes prohibitive when the global dataset $\mathcal{D}$ is large. Furthermore, in case we adopt a centralized network topology where each agent i transmits its local dataset $\mathcal{D}_i$ to a central server, several issues arise. These include concerns over security and robustness, traffic network congestion; and privacy [42]. In addition, in autonomous vehicles and teams of robots, agents may encounter communication range limitations due their distance. With these challenges in mind, we make the following assumptions.

Assumption 2. *Each agent i communicates information to its one-hop neighbors $j \in \mathcal{N}_i$ without sharing raw data.*

Assumption 3. *Every agent i can develop a local sub-model $\mathcal{M}_i$ from its local dataset $\mathcal{D}_i$ that is statistically independent.*

Assumption 3 is common in distributed optimization [64], [65]. In particular, the global marginal likelihood is approximated as the product of local marginal likelihoods that yields,

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

where the local marginal likelihood of agent i is represented by $p_i(\boldsymbol{y}_i \mid \boldsymbol{X}_i) \sim \mathcal{N}(0, \boldsymbol{C}_{\theta,i})$ with local covariance matrix $\boldsymbol{C}_{\theta,i} = \boldsymbol{K}_i + \sigma_\epsilon^2 I_{N_i}$ and $\boldsymbol{K}_i = k(\boldsymbol{X}_i, \boldsymbol{X}_i) \in \mathbb{R}^{N_i \times N_i}$. Assumption 3 and (5), imply the approximation of the inverse covariance matrix with the inverse of local block diagonal covariances $\boldsymbol{C}_\theta^{-1} \approx \mathrm{diag}\{\boldsymbol{C}_{\theta,1}^{-1}, \boldsymbol{C}_{\theta,2}^{-1}, \ldots, \boldsymbol{C}_{\theta,M}^{-1}\}$. Provided the hyperparameters $\hat{\boldsymbol{\theta}}$, aggregation of GP experts perform joint predictions with local data. Each local agent develops a local GP model $\mathcal{M}_i$ using its local dataset $\mathcal{D}_i$. Joint prediction is performed by sharing the local GP models $p_i(y_* \mid \mathcal{D}_i, \boldsymbol{x}_i) \sim \mathcal{GP}(\mu_i(\boldsymbol{x}_i), \sigma_i^2(\boldsymbol{x}_i))$ that are characterized by a local mean and variance,

$$\mu_i(\boldsymbol{x}_*) = \boldsymbol{k}_{*,i}^\intercal \boldsymbol{C}_{\theta,i}^{-1} \boldsymbol{y}_i, \tag{6}$$
$$\sigma_i^2(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_{*,i}^\intercal \boldsymbol{C}_{\theta,i}^{-1} \boldsymbol{k}_{*,i}, \tag{7}$$

where $\boldsymbol{k}_{*,i} = k(\boldsymbol{x}_*, \boldsymbol{X}_i) \in \mathbb{R}^{N_i}$.

Definition 1. *[36] An aggregate GP method with joint prediction mean $\mu_{\mathcal{A}}$, variance $\sigma_{\mathcal{A}}^2$, full GP prediction mean μ_{full}, and variance σ_{full}^2 is consistent if,*

$$\lim_{N\to\infty} \mu_{\text{full}}(\boldsymbol{x}_*) - \mu_{\mathcal{A}}(\boldsymbol{x}_*) \to 0, \quad \forall \boldsymbol{x}_*,$$
$$\lim_{N\to\infty} \sigma_{\text{full}}^2(\boldsymbol{x}_*) - \sigma_{\mathcal{A}}^2(\boldsymbol{x}_*) \to 0, \quad \forall \boldsymbol{x}_*,$$

where subscript $\mathcal{A}$ denotes then aggregation method.

Definition 1 implies that as the dataset size tends to infinity, the aggregated prediction mean and variance are identical to the prediction mean and variance of full GP (3).

Problem 1. *Implement GP prediction in a one-hop decentralized network of agents (Assumption 1), where the agents can be grouped into subteams of neighboring units with limited computations and communication (Assumption 1, 2). Provide formal proofs for the proposed decentralized methods, demonstrating that they maintain the consistency properties (Definition 1) of centralized topologies.*

The primary focus of Problem 1 is on the implementation of GP prediction in decentralized teams of agents. We impose the one-hop line graph topology to provide a lower bound for the proposed methods. By analyzing the worst case scenario in terms of network connectivity, we anticipate our methods to perform more efficiently in topologies with higher connectivity. Problem 1 addresses a broad range of applications, from no data exchange to enable federated learning, to partial data exchange that scale to larger networks. In addition, Problem 1 involves the joint prediction with subteams of agents, allowing the formulation of covariance-based nearest neighbors. To this end, we aim to reduce the communication from distant entities with insignificant statistic correlation.

III. Existing Centralized GP Prediction

In this section, we discuss existing centralized GP prediction methods: product of experts (PoE) [31], generalized PoE (gPoE) [33], Bayesian Committee Machine (BCM) [30], robust BCM (rBCM) [32], generalized robust BCM (grBCM) [35], and nested pointwise aggregation of experts (NPAE) [34], [36].

A. Product of Experts (PoE) Family

After computing the local mean (6) and variance (7), the joint mean and precision of PoE and gPoE yields,

$$\mu_{\text{(g)PoE}}(\boldsymbol{x}_*) = \sigma_{\text{(g)PoE}}^2(\boldsymbol{x}_*) \sum_{i=1}^M \beta_i \sigma_i^{-2}(\boldsymbol{x}_*) \mu_i(\boldsymbol{x}_*), \tag{8}$$

$$\sigma_{\text{(g)PoE}}^{-2}(\boldsymbol{x}_*) = \sum_{i=1}^M \beta_i \sigma_i^{-2}(\boldsymbol{x}_*), \tag{9}$$

where $\beta_i = 1$ for PoE, and $\beta_i = 1/M$ for gPoE. The original gPoE [33] considers weight β_i to be the difference in differential entropy, but it limits the computational graph to have a single layer. To allow multiple layer GP aggregation, an average weight β_i is proposed in [32]. It is empirically observed that for a disjoint partition of local datasets $\mathcal{D}_i$, both PoE and gPoE produce inconsistent predictions.

Proposition 1. *[35, Proposition 1] For a disjoint partition of local datasets $\mathcal{D}_i$, the constant weight of PoE results in overconfident joint variance as the number of observations N tends to infinity, i.e.* $\lim_{N\to\infty} \sigma_{PoE}^2(\boldsymbol{x}_*) \to 0$. *The average weight of gPoE produces a conservative, yet finite joint variance as the number of observations N tends to infinity, i.e. $\sigma_{full}^2 < \lim_{N\to\infty} \sigma_{gPoE}^2(\boldsymbol{x}_*) < \sigma_{**}^2$, where σ_{full}^2 is the target variance of a full GP and σ_{**}^2 the prior variance.*

Proposition 2. *[Appendix A] The PoE and gPoE make identical mean predictions* (8).

The local time computational complexity of both PoE and gPoE is governed by the local variance (7), that is $\mathcal{O}(N_i^2) = \mathcal{O}(N^2/M^2)$ for the multiplication of the quadratic term. Both the PoE and gPoE alleviates the computations compared to the full GP $\mathcal{O}(N^2)$. The space complexity requires $\mathcal{O}(N_i^2 + DN_i) = \mathcal{O}(N^2/M^2 + D(N/M))$ capacity to store the inverse of the local inverted covariance matrix $\boldsymbol{C}_{\theta,i}^{-1}$ and the vector of local observations $\boldsymbol{y}_i$. Thus, the space requirement is relaxed compared to full GP that occupies $\mathcal{O}(N^2 + DN)$ memory. The total communication complexity from all entities to the central node is $\mathcal{O}(2M)$ to transmit all local mean μ_i and local variance σ_i^2 values. A comparison of PoE and gPoE is presented in Table 2.

B. Bayesian Committee Machine (BCM) Family

The BCM, rBCM, and grBCM make the following additional assumption other than Assumption 3.

Assumption 4. *The dataset of every agent i is conditionally independent from any other dataset of agent $j \neq i$ given the posterior distribution f_*, i.e. $\mathcal{D}_i \perp\!\!\!\perp \mathcal{D}_j \mid f_*$.*

After computing the local mean (6) and the local variance (7), the joint mean and precision of the BCM and rBCM $\mathcal{M}_{\mathcal{A}} = \mathcal{M}_{\text{(r)BCM}}$ is provided by,

$$\mu_{\text{(r)BCM}}(\boldsymbol{x}_*) = \sigma^2_{\text{(r)BCM}}(\boldsymbol{x}_*) \sum_{i=1}^{M} \beta_i \sigma_i^{-2}(\boldsymbol{x}_*) \mu_i(\boldsymbol{x}_*), \tag{10}$$

$$\sigma^{-2}_{\text{(r)BCM}}(\boldsymbol{x}_*) = \sum_{i=1}^{M} \beta_i \sigma_i^{-2}(\boldsymbol{x}_*) + \left(1 - \sum_{i=1}^{M} \beta_i\right) \sigma_{**}^{-2}, \tag{11}$$

where $\beta_i = 1$ for BCM, $\beta_i = 0.5[\log \sigma_{**}^2 - \log \sigma_i^2(\boldsymbol{x}_*)]$, and the prior variance $\sigma_{**}^2 = k_{**} + \sigma_\epsilon^2$ for rBCM. For rBCM β_i describes the difference in the differential entropy between the prior and the posterior distribution. It is empirically observed that for a disjoint partition of local datasets $\mathcal{D}_i$, both BCM and rBCM produce inconsistent mean predictions. However, the joint mean of BCM and rBCM converges to the prior mean slower than the PoE and gPoE.

Proposition 3. *[35, Proposition 1] For a disjoint partition of local datasets $\mathcal{D}_i$, the BCM and rBCM result in overconfident joint variance as the number of observations N tends to infinity, i.e. $\lim_{N\to\infty} \sigma^2_{\text{(r)BCM}}(\boldsymbol{x}_*) \to 0$.*

The time and space complexity of BCM is the same to PoE and gPoE. In addition, BCM requires similar communications with the PoE family. However, the rBCM entails $\mathcal{O}(3M)$ communication complexity to exchange the local mean μ_i, the local variance σ_i^2, and the difference in the differential entropy between the prior and posterior distribution β_i. Note that β_i in rBCM can be computed by the central node and recovers the communication complexity of PoE and gPoE. Yet, we prefer to express β_i as part of the communication exchange, because in the ensuing discussion the central node is removed. A comparison of BCM and rBCM with other aggregation methods is illustrated in Table 2.

Assumption 5. *Each agent i communicates information to its one-hop neighbors $j \in \mathcal{N}_i$ with partial sharing of data.*

In Assumption 2, we restrict sharing of any raw data among agents, while Assumption 5 permits partial information exchange of local datasets. We relax Assumption 2 to enable predictions in larger multi-agent systems at the cost of violating federated learning requirements. The main ideas of grBCM is to equip every agent with a new dataset that has global information on the underlying latent function and ensure consistency (Definition 1). 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 5) to compose the *communication dataset* $\mathcal{D}_\text{c} = \{\mathcal{D}_{\text{-i}}\}_{i=1}^M = \{\boldsymbol{X}_\text{c}, \boldsymbol{y}_\text{c}\}$. 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 that includes the local dataset $\mathcal{D}_i$ and the communication dataset $\mathcal{D}_\text{c}$.

The agents use the local augmented dataset $\mathcal{D}_{+i}$ to compute the augmented local mean μ_{+i} (6) and the augmented local variance σ_{+i}^2 (7). In addition, grBCM requires the computation of the communication local mean μ_c (6) and the communication local variance σ_c^2 (7) using exclusively the communication dataset $\mathcal{D}_\text{c}$. The joint mean and precision of grBCM $\mathcal{M}_{\mathcal{A}} = \mathcal{M}_{\text{grBCM}}$ yield,

$$\mu_{\text{grBCM}}(\boldsymbol{x}_*) = \sigma^2_{\text{grBCM}}(\boldsymbol{x}_*) \left(\sum_{i=1}^{M} \beta_i \sigma_{+i}^{-2}(\boldsymbol{x}_*) \mu_{+i}(\boldsymbol{x}_*) - \left(\sum_{i=1}^{M} \beta_i - 1 \right) \sigma_\text{c}^{-2}(\boldsymbol{x}_*) \mu_\text{c}(\boldsymbol{x}_*) \right), \tag{12}$$

$$\sigma^{-2}_{\text{grBCM}}(\boldsymbol{x}_*) = \sum_{i=1}^{M} \beta_i \sigma_{+i}^{-2}(\boldsymbol{x}_*) + \left(1 - \sum_{i=1}^{M} \beta_i\right) \sigma_\text{c}^{-2}(\boldsymbol{x}_*), \tag{13}$$

where $\beta_1 = 1$, $\beta_i = 1/2[\log \sigma_\text{c}^2(\boldsymbol{x}_*) - \log \sigma_{+i}^2(\boldsymbol{x}_*)]$, $i > 2$.

Proposition 4. *[35, Proposition 3] For any collection of aggregated prediction mean values $\mu_1(\boldsymbol{x}_*), \dots, \mu_M(\boldsymbol{x}_*)$ the grBCM is consistent.*

The local time complexity for grBCM includes the computation of the augmented local variance σ_{+i}^2, the local communication variance σ_c^2, and the inversion of the communication dataset covariance matrix $\boldsymbol{K}_\text{c} = k(\boldsymbol{X}_\text{c}, \boldsymbol{X}_\text{c})$. That is $\mathcal{O}((2N_i)^2 + N_i^2 + N_i^3) = \mathcal{O}(N^2/M^2(5 + N/M))$. Then, the inverse of the local augmented covariance matrix $\boldsymbol{C}_{\theta,+i}^{-1}$ and the local augmented dataset $\mathcal{D}_{+i}$ occupy $\mathcal{O}((2N_i)^2 + D(2N_i)) = \mathcal{O}(2(N^2/M^2 + DN/M) + 2N^2/M^2)$ space. The

TABLE 2. Time, Space, and Communication Complexity for Centralized GP Aggregated Prediction

		FULL-GP	(g)PoE & BCM	rBCM	grBCM	NPAE
Local	Time	-	$\mathcal{O}(\zeta)$	$\mathcal{O}(\zeta)$	$\mathcal{O}((5+N/M)\zeta)$	$\mathcal{O}(N\zeta)$
	Space	-	$\mathcal{O}(\xi)$	$\mathcal{O}(\xi)$	$\mathcal{O}(2\xi+2(N^2/M^2))$	$\mathcal{O}(\xi+DN)$
Global	Time	$\mathcal{O}(N^2)$	$\mathcal{O}(M)$	$\mathcal{O}(M)$	$\mathcal{O}(M)$	$\mathcal{O}(M^3)$
	Space	$\mathcal{O}(N^2+DN)$	$\mathcal{O}(2M)$	$\mathcal{O}(3M)$	$\mathcal{O}(5M)$	$\mathcal{O}(M^2)$
	Comm	-	$\mathcal{O}(2M)$	$\mathcal{O}(3M)$	$\mathcal{O}(5M)$	$\mathcal{O}(M^2)$

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

total communications from all entities to the central node is $\mathcal{O}(5M)$ to transmit all local augmented means μ_{+i}, local augmented variances σ^2_{+i}, local communication means μ_{c}, local communication variances σ^2_{c}, and all differences in the differential entropy β_i to the central node. A comparison of grBCM with other aggregation methods is shown in Table 2.

C. Nested Pointwise Aggregation of Experts (NPAE)

The main idea of NPAE is to use covariance between sub-models $\mathcal{M}_i$. The local computations of NPAE for agent i include: i) local prediction mean $\mu_i(\boldsymbol{x}_*) \in \mathbb{R}$ (6); ii) i-th entry of the cross-covariance vector $[\boldsymbol{k}_A(\boldsymbol{x}_*)]_i \in \mathbb{R}$; and iii) i-th row of the covariance $\text{row}_i\{\boldsymbol{C}_{\theta,A}(\boldsymbol{x}_*)\} \in \mathbb{R}^M$. Thus, NPAE requires the local computation of two additional quantities other than (6). These are the cross-covariance and the covariance for each agent i,

$$[\boldsymbol{k}_A(\boldsymbol{X}_i,\boldsymbol{x}_*)]_i = \boldsymbol{k}_{i,*}^\intercal \boldsymbol{C}_{\theta,i}^{-1}\boldsymbol{k}_{i,*}, \tag{14}$$

$$[\text{row}_i\{\boldsymbol{C}_{\theta,A}(\boldsymbol{X}_i,\boldsymbol{X}_j,\boldsymbol{x}_*)\}]_j = \boldsymbol{k}_{i,*}^\intercal \boldsymbol{C}_{\theta,i}^{-1}\boldsymbol{C}_{\theta,ij}\boldsymbol{C}_{\theta,j}^{-1}\boldsymbol{k}_{j,*}, \tag{15}$$

where $\boldsymbol{C}_{\theta,ij} = k(\boldsymbol{X}_i,\boldsymbol{X}_j) + \sigma_\epsilon^2 I_{N_i} \in \mathbb{R}^{N_i \times N_i}$, $\boldsymbol{k}_{i,*} = (\boldsymbol{X}_i,\boldsymbol{x}_*) \in \mathbb{R}^{N_i}$, and $\boldsymbol{k}_{j,*} = (\boldsymbol{X}_j,\boldsymbol{x}_*) \in \mathbb{R}^{N_i}$ for all $j \neq i$. The next step is to aggregate the local sub-models and obtain the joint prediction mean and variance,

$$\mu_{\text{NPAE}}(\boldsymbol{x}_*) = \boldsymbol{k}_A^\intercal \boldsymbol{C}_{\theta,A}^{-1}\boldsymbol{\mu}, \tag{16}$$

$$\sigma^2_{\text{NPAE}}(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_A^\intercal \boldsymbol{C}_{\theta,A}^{-1}\boldsymbol{k}_A, \tag{17}$$

where $\boldsymbol{C}_{\theta,A} = \{\text{row}_i\{\boldsymbol{C}_{\theta,A}\}\}_{i=1}^M \in \mathbb{R}^{M\times M}$, $\boldsymbol{k}_A = \{\boldsymbol{k}_A(\boldsymbol{X}_i,\boldsymbol{x}_*)\}_{i=1}^M \in \mathbb{R}^M$, and $\boldsymbol{\mu} = \{\mu_i\}_{i=1}^M \in \mathbb{R}^M$.

Proposition 5. *[34, Proposition 2] For any collection of aggregated prediction mean values $\mu_1(\boldsymbol{x}_*),\ldots,\mu_M(\boldsymbol{x}_*)$ the NPAE is consistent.*

The local time complexity for NPAE is governed by the computation of all local inverted covariance matrices for every other agent $\boldsymbol{C}_{\theta,j}^{-1}$, $j \neq i$ (14) which yields $\mathcal{O}((M-1)N_i^3) = \mathcal{O}(N^3/M^2)$ computations. Additionally, the aggregated covariance $\boldsymbol{C}_{\theta,A}$ needs to be inverted on the central node that entails $\mathcal{O}(M^3)$ computations. The local memory footprint is $\mathcal{O}(N_i^2 + DN_i + MDN_i) = \mathcal{O}(N^2/M^2 + D(N/M) + DN)$, including the local inverted covariance matrix $\boldsymbol{C}_{\theta,i}^{-1}$, the local dataset $\mathcal{D}_i$, and the inputs of all other datasets $\boldsymbol{X}_j$ for all $j \neq i$. The total communication complexity yields $\mathcal{O}((M+1)M) = \mathcal{O}(M^2)$ governed by the transmission of the row aggregated covariance $\text{row}_i\{\boldsymbol{C}_{\theta,A}\}$, for all $i \in \mathcal{V}$. A major disadvantage of NPAE is the high global time complexity, yet in the ensuing discussion we distribute the computation among agents by using decentralized iterative techniques. A comparison of NPAE with other aggregation methods is provided in Table 2.

IV. Proposed Decentralized GP Prediction

In this section, we review three foundational decentralized algorithms: i) the discrete-time average consensus (DAC) method [6]; ii) the Jacobi over-relaxation method (JOR) [5, Ch. 2.4]; and iii) the power method (PM) [61, Chapter 8]. We introduce seven decentralized aggregation methods built using elements from the foundational algorithms, provide a consistency analysis, and discuss their computational, space, and communication complexity.

A. Foundational Decentralized Algorithms

1) Discrete-Time Average Consensus (DAC)

The DAC is an iterative and parallel method to compute the average of a vector $\boldsymbol{w} \in \mathbb{R}^M$ within a network. More specifically, every agent i has access to one element $w_i \in \mathbb{R}$ and the goal is to compute the average $\bar{\boldsymbol{w}} = (1/M)\sum_{i=1}^M w_i$. The DAC update law yields,

$$w_i^{(s+1)} = w_i^{(s)} + \epsilon \sum_{j\in\mathcal{N}_i} a_{ij}\left(w_j^{(s)} - w_i^{(s)}\right), \tag{18}$$

where ϵ is the parameter of the Perron matrix and a_{ij} is the (i,j)-th entry of the adjacency matrix. Use of consensus protocols inherently requires that each node can independently determine convergence. In other words, an agent's individual convergence does not guarantee that all agents have reached consensus. We employ a maximin stopping criterion [66] to locally detect convergence across the network. An additional assumption is required to implement the DAC.

Assumption 6. *The network size M is known to all agents.*

Lemma 1. *[6, Theorem 2], [67, Corollary 5.2] Let Assumption 1 hold. If $\epsilon \in (0, 1/\Delta)$, then the DAC (18) converges to the average $\bar{\boldsymbol{w}}$ for any initialization $w_i^{(0)}$ with convergence time $T_M(\epsilon) = \mathcal{O}(M^3\log(M/\epsilon))$.*

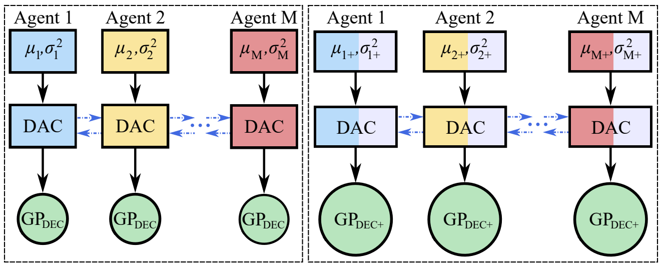

FIGURE 2. The structure of the proposed DEC-PoE and DEC-BCM families. Blue dotted lines correspond to communication (connected). Every agent implements discrete-time average consensus (DAC) methods. The gray color of the squares in (b) indicates the formulation of local augmented datasets.

TABLE 3. Communication Complexity of Decentralized GP Aggregations

Method	Graph	Communication Complexity
DEC-PoE	C	$\mathcal{O}(2\chi)$
DEC-gPoE	C	$\mathcal{O}(2\chi)$
DEC-BCM	C	$\mathcal{O}(2\chi)$
DEC-rBCM	C	$\mathcal{O}(3\chi)$
DEC-grBCM	C	$\mathcal{O}(3\chi)$
DEC-NPAE [58]	CC	$\mathcal{O}(2Ms_{\text{JOR}}^{\text{end}} + 2\chi + M\xi)$
DEC-NPAE*	CC	$\mathcal{O}(2M(s_{\text{JOR}^\star}^{\text{end}} + s_{\text{PM}}^{\text{end}}) + 2\chi + M\xi + M^2)$

$\chi = s_{\text{DAC}}^{\text{end}}\text{card}(\mathcal{N}_i)$, $\xi = N^2/M^2 + D(N/M)$, C: connected, CC: complete connected.

2) Jacobi Over-Relaxation (JOR)

The JOR is an iterative and parallel method to solve a system of linear algebraic equations in the form of $\boldsymbol{H}\boldsymbol{q} = \boldsymbol{b}$, where $\boldsymbol{H} = [h_{ij}] \in \mathbb{R}^{M\times M}$ is a known non-singular matrix with non-zero diagonal entries $h_{ii} \neq 0$, $\boldsymbol{b} \in \mathbb{R}^M$ is a known vector, and $\boldsymbol{q} \in \mathbb{R}^M$ is an unknown vector. In particular, the i-th node knows: i) the i-th row of the known matrix $\text{row}_i\{\boldsymbol{H}\} \in \mathbb{R}^{1\times M}$; and ii) the i-th element of the known vector $b_i \in \mathbb{R}$. The objective is to find $q_i \in \mathbb{R}$, the i-th element of the unknown $\boldsymbol{q}$. The JOR iterative scheme yields,

$$q_i^{(s+1)} = (1-\omega)q_i^{(s)} + \frac{\omega}{h_{ii}}\left(b_i - \sum_{j\neq i} h_{ij}q_j^{(s)}\right), \tag{19}$$

where $\omega \in (0,1)$ the relaxation parameter.

Remark 1. *The summation in* (19) *requires communication with all agents, as it is computed over $j \neq i$. This means that each agent must know the update value* (19) *of every other agent $\{\boldsymbol{q}_j^{(s)}\}_{j\neq i}$. That is a major restriction, because it imposes a complete graph topology. Although JOR is used for distributed networks [43], [44], [68], it is unrealistic for many applications due to communication restrictions.*

We evaluate the use of JOR, as complete networks are feasible in applications with small fleet size. For non-complete network topologies, distributed flooding is required at every iteration to obtain $\{\boldsymbol{q}_j^{(s)}\}_{j\neq i}$ and implement (19). The number of inter-agent communications for distributed flooding is the diameter of the graph $\text{diam}(\mathcal{G})$ with total number of iterations $s_{\text{JOR}} = \text{diam}(\mathcal{G})s_{\text{JOR}}^{\text{end}}$. In the following Section we present the distributed algorithm to solve systems of linear equations (DALE), which overcomes the limitations of the JOR method.

Lemma 2. *[60, Theorem 2] Let the graph $\mathcal{G}$ be complete. If $\boldsymbol{H}$ is symmetric and PD, and $\omega < 2/M$, then the JOR converges to the solution for any initialization $q_i^{(0)}$.*

Lemma 3. *[60, Theorem 4] Let the graph $\mathcal{G}$ be complete. If $\boldsymbol{H}$ is symmetric and PD, and $\omega^\star = 2/(\overline{\lambda}(\boldsymbol{R}) + \underline{\lambda}(\boldsymbol{R}))$ where $\boldsymbol{R} = \text{diag}(\boldsymbol{H})^{-1}\boldsymbol{H}$, then the JOR converges to the solution for any initialization $q_i^{(0)}$ with the optimal rate.*

Remark 2. *The difference between Lemma 2 and 3 is that the latter employs the optimal relaxation factor $\omega^\star$ which is characterized by the eigenvalues of $\boldsymbol{R}$. In principle, the smaller the relaxation factor ω the slower the convergence speed [69]. Since $\omega^\star > 2/M$, the optimal relaxation leads to faster convergence of JOR to the solution. To compute $\omega^\star$ in a network of agents, additional communication is required to distributively estimate the maximum and minimum eigenvalues of $\boldsymbol{R}$. However, the sufficient condition for ω of Lemma 2 can be locally computed with no communication. The distributed method for the computation of $\omega^\star$ entails $s_{\omega^\star}^{end}$ iterations, JOR with ω from Lemma 2 converges after s_{JOR}^{end} iterations, and JOR with $\omega^\star$ from Lemma 3 converges after $s_{JOR^\star}^{end}$ iterations. Then, $\omega^\star$ is communication-wise more efficient in decentralized networks when $s_{\omega^\star}^{end} + s_{JOR^\star}^{end} < s_{JOR}^{end}$.*

3) Power Method (PM)

The optimal relaxation factor $\omega^\star$ involves the maximum eigenvalue $\overline{\lambda}(\boldsymbol{R})$ and minimum eigenvalue $\underline{\lambda}(\boldsymbol{R})$ (Lemma 3). We employ the power method (PM) to compute $\overline{\lambda}(\boldsymbol{R})$ and the inverse power method (IPM) to compute $\underline{\lambda}(\boldsymbol{R})$. The PM is a two step iterative algorithm that follows,

$$\boldsymbol{g}^{(s+1)} = \boldsymbol{R}\boldsymbol{e}^{(s)} \tag{20a}$$

$$\boldsymbol{e}^{(s+1)} = \frac{1}{\|\boldsymbol{g}^{(s+1)}\|_\infty}\boldsymbol{g}^{(s+1)}, \tag{20b}$$

where $\|\cdot\|_\infty$ denotes the infinity norm. As the PM algorithm converges $\|\boldsymbol{e}^{(s)} - \boldsymbol{e}^{(s-1)}\|_2 \to 0$, the infinity norm approximates the dominant eigenvalue $\|\boldsymbol{g}^{(s)}\|_\infty \approx \overline{\lambda}(\boldsymbol{R})$. After obtaining $\overline{\lambda}(\boldsymbol{R})$, we formulate the spectral shift of $\boldsymbol{R}$, that is $\boldsymbol{B} = \boldsymbol{R} - \overline{\lambda}(\boldsymbol{R})I_M$. The IPM is the application of PM (20) on $\boldsymbol{B}$. Next, we compute the minimum eigenvalue as $\underline{\lambda}(\boldsymbol{R}) = |\overline{\lambda}(\boldsymbol{B}) - \overline{\lambda}(\boldsymbol{R})|$. In order to obtain both $\overline{\lambda}(\boldsymbol{R})$ and $\underline{\lambda}(\boldsymbol{R})$, we need to execute the PM algorithm (20) two sequential times. Let the first PM algorithm to converge after $s_{\text{PM}}^{\text{end}}$ iterations and the second after $s_{\text{IPM}}^{\text{end}}$ iterations. The use of

Algorithm 1 DEC-PoE

```
Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, $k$, $M$, $\boldsymbol{x}_*$, $\Delta$
Output: $\mu_{\text{DEC-PoE}}$, $\sigma_{\text{DEC-PoE}}^{-2}$
 1: $\epsilon = 1/\Delta$
 2: for each $i \in \mathcal{V}$ do
 3:     $\mu_i \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}_{\theta,i}^{-1})$ (6)
 4:     $\sigma_i^{-2} \leftarrow \texttt{localVariance}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}_{\theta,i}^{-1},)$ (7)
 5:     initialize $w_{\mu,i}^{(0)} = \beta_i \sigma_i^{-2} \mu_i$, $w_{\sigma^{-2},i}^{(0)} = \beta_i \sigma_i^{-2}$, $\beta_i = 1$
 6:     repeat
 7:         communicate $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$ to agents in $\mathcal{N}_i$
 8:         $w_{\mu,i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\mu,i}^{(s)}, \{\boldsymbol{w}_{\mu,j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)   ▷ DAC1
 9:         $w_{\sigma^{-2},i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\sigma^{-2},i}^{(s)}, \{\boldsymbol{w}_{\sigma^{-2},j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)   ▷ DAC2
10:     until maximin stopping criterion
11:     $\sigma_{\text{DEC-PoE}}^{-2} = M w_{\sigma^{-2},i}^{(\text{end})}$ (9)
12:     $\mu_{\text{DEC-PoE}} = \sigma_{\text{DEC-PoE}}^{2} M w_{\mu,i}^{(\text{end})}$ (8)
13: end for
```

Algorithm 2 DEC-gPoE

```
Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, $k$, $M$, $\boldsymbol{x}_*$, $\Delta$
Output: $\mu_{\text{DEC-gPoE}}$, $\sigma_{\text{DEC-gPoE}}^{-2}$
 1: Same to Algorithm 1 with $\beta_i = 1/M$ instead of $\beta_i = 1$ (line 5)
```

the optimal relaxation is communication-wise more efficient if $s_{\text{PM}}^{\text{end}} + s_{\text{IPM}}^{\text{end}} + s_{\text{JOR}^\star}^{\text{end}} < s_{\text{JOR}}^{\text{end}}$ (Remark 2).

Lemma 4. *[61, Chapter 8] Let the graph $\mathcal{G}$ be complete. If $\boldsymbol{H}$ is symmetric, then the PM converges to the dominant real eigenvalue $\overline{\lambda}(\boldsymbol{R})$ with convergence rate $\mathcal{O}((\lambda_2/\overline{\lambda})^{s_{PM}^{end}})$, where λ_2 is the second largest eigenvalue.*

B. Proposed Decentralized Product of Experts

The decentralized PoE (DEC-PoE) method makes use of two DAC algorithms (Figure 2-(a)). The first DAC computes the average $(1/M)\sum_{i=1}^{M} \beta_i \sigma_i^{-2}$ and the second $(1/M)\sum_{i=1}^{M} \beta_i \sigma_i^{-2}\mu_i$, where $\beta_i = 1$. At every iteration of DAC each agent communicates both computed values $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$ to its neighbors $\mathcal{N}_i$. After convergence, each DAC average is multiplied by the number of nodes M and follow (8), (9) to recover the DEC-PoE mean and precision (Algorithm 1). The time and space complexity are the same to the local time and space complexity of the PoE family (Table 2). Let $s_{\text{DAC}}^{\text{end}}$ be the maximum number of iterations of the two DAC to converge. The total communications are $\mathcal{O}(2 s_{\text{DAC}}^{\text{end}} \text{card}(\mathcal{N}_i))$ (Table 3).

Next, we form the decentralized gPoE (DEC-gPoE) (Figure 2-(a)). The DEC-gPoE is similar to the DEC-PoE, but $\beta_i = 1/M$ instead of $\beta_i = 1$ (Algorithm 2). The time, space, and communication complexity are the same to DEC-PoE.

C. Proposed Decentralized Bayesian Committee Machine

The decentralized BCM (DEC-BCM) method employs two DAC algorithms (Figure 2-(a)). The first DAC computes the average $(1/M)\sum_{i=1}^{M} \beta_i \sigma_i^{-2}$ and the second $(1/M)\sum_{i=1}^{M} \beta_i \sigma_i^{-2}\mu_i$, where $\beta_i = 1$. At every iteration of DAC each agent communicates both computed values $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$ to its neighbors $\mathcal{N}_i$. After convergence, each average is multiplied by the number of nodes M and follow (10), (11) to recover the DEC-BCM mean and precision (Algorithm 3). The time, space, and communication complexity are the same to DEC-PoE family.

We introduce the decentralized rBCM (DEC-rBCM) technique that employs three DAC algorithms to compute the averages $(1/M)\sum_{i=1}^{M} \beta_i \sigma_i^{-2}$, $(1/M)\sum_{i=1}^{M} \beta_i \sigma_i^{-2}\mu_i$, and $(1/M)\sum_{i=1}^{M} \beta_i$, where $\beta_i = 0.5[\log \sigma_{**}^2 - \log \sigma_i^2]$. At every iteration of DAC each agent communicates $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$, $w_{\beta_i}^{(s)}$ to its neighbors $\mathcal{N}_i$. After convergence, each average

Algorithm 3 DEC-BCM

```
Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, $k$, $M$, $\boldsymbol{x}_*$, $\Delta$
Output: $\mu_{\text{DEC-BCM}}$, $\sigma_{\text{DEC-BCM}}^{-2}$
 1: $\epsilon = 1/\Delta$
 2: for each $i \in \mathcal{V}$ do
 3:     $\mu_i \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}_{\theta,i}^{-1})$ (6)
 4:     $\sigma_i^{-2} \leftarrow \texttt{localVariance}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}_{\theta,i}^{-1},)$ (7)
 5:     $\sigma_{**}^2 = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma_\epsilon^2$
 6:     initialize $w_{\mu,i}^{(0)} = \beta_i \sigma_i^{-2} \mu_i$, $w_{\sigma^{-2},i}^{(0)} = \beta_i \sigma_i^{-2}$, $\beta_i = 1$
 7:     repeat   ▷ 2×DAC
 8:         communicate $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$ to agents in $\mathcal{N}_i$
 9:         $w_{\mu,i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\mu,i}^{(s)}, \{\boldsymbol{w}_{\mu,j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
10:         $w_{\sigma^{-2},i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\sigma^{-2},i}^{(s)}, \{\boldsymbol{w}_{\sigma^{-2},j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
11:     until maximin stopping criterion
12:     $\sigma_{\text{DEC-BCM}}^{-2} = M w_{\sigma^{-2},i}^{(\text{end})} + (1 - \sum_{i=1}^{M} \beta_i)\sigma_{**}^{-2}$ (11)
13:     $\mu_{\text{DEC-BCM}} = \sigma_{\text{DEC-BCM}}^{2} M w_{\mu,i}^{(\text{end})}$ (10)
14: end for
```

Algorithm 4 DEC-rBCM

```
Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, $k$, $M$, $\boldsymbol{x}_*$, $\Delta$
Output: $\mu_{\text{DEC-rBCM}}$, $\sigma_{\text{DEC-rBCM}}^{-2}$
 1: $\epsilon = 1/\Delta$
 2: for each $i \in \mathcal{V}$ do
 3:     $\mu_i \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}_{\theta,i}^{-1})$ (6)
 4:     $\sigma_i^{-2} \leftarrow \texttt{localVariance}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}_{\theta,i}^{-1})$ (7)
 5:     $\sigma_{**}^2 = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma_\epsilon^2$
 6:     initialize $w_{\mu,i}^{(0)} = \beta_i \sigma_i^{-2} \mu_i$, $w_{\sigma^{-2},i}^{(0)} = \beta_i \sigma_i^{-2}$, $w_{\beta_i}^{(0)} = \beta_i$, $\beta_i = 0.5[\log \sigma_{**}^2 - \log \sigma_i^2]$
 7:     repeat   ▷ 3×DAC
 8:         communicate $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$, $w_{\beta_i}^{(s)}$ to agents in $\mathcal{N}_i$
 9:         $w_{\mu,i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\mu,i}^{(s)}, \{\boldsymbol{w}_{\mu,j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
10:         $w_{\sigma^{-2},i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\sigma^{-2},i}^{(s)}, \{\boldsymbol{w}_{\sigma^{-2},j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
11:         $w_{\beta_i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\beta_i}^{(s)}, \{\boldsymbol{w}_{\beta_j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
12:     until maximin stopping criterion
13:     $\sigma_{\text{DEC-rBCM}}^{-2} = M w_{\sigma^{-2},i}^{(\text{end})} + (1 - M w_{\beta_i}^{(\text{end})})\sigma_{**}^{-2}$ (11)
14:     $\mu_{\text{DEC-rBCM}} = \sigma_{\text{DEC-rBCM}}^{2} M w_{\mu,i}^{(\text{end})}$ (10)
15: end for
```

Algorithm 5 DEC-grBCM

```
Input: $\mathcal{D}_{+i}(\boldsymbol{X}_{+i}, \boldsymbol{y}_{+i})$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,+i}^{-1}$, $\mathcal{N}_i$, $k$, $M$, $\boldsymbol{x}_*$, $\Delta$
Output: $\mu_{\text{DEC-grBCM}}$, $\sigma_{\text{DEC-grBCM}}^{-2}$
 1: $\epsilon = 1/\Delta$
 2: for each $i \in \mathcal{V}$ do
 3:     $\mu_{+i} \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_{+i}, \boldsymbol{C}_{\theta,+i}^{-1})$ (6)
 4:     $\sigma_{+i}^{-2} \leftarrow \texttt{localVariance}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_{+i}, \boldsymbol{C}_{\theta,+i}^{-1})$ (7)
 5:     $\sigma_{\mathrm{c}}^2 = k(\boldsymbol{X}_{\mathrm{c}}, \boldsymbol{X}_{\mathrm{c}})$
 6:     initialize $w_{\mu,i}^{(0)} = \beta_i \sigma_{+i}^{-2} \mu_{+i}$, $w_{\sigma^{-2},i}^{(0)} = \beta_i \sigma_{+i}^{-2}$, $w_{\beta_i}^{(0)} = \beta_i$, $\beta_i = 0.5[\log \sigma_{\mathrm{c}}^2 - \log \sigma_{+i}^2]$
 7:     repeat                                                    ▷ 3×DAC
 8:         communicate $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$, $w_{\beta_i}^{(s)}$ to agents in $\mathcal{N}_i$
 9:         $w_{\mu,i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\mu,i}^{(s)}, \{\boldsymbol{w}_{\mu,j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
10:         $w_{\sigma^{-2},i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\sigma^{-2},i}^{(s)}, \{\boldsymbol{w}_{\sigma^{-2},j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
11:         $w_{\beta_i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\beta_i}^{(s)}, \{\boldsymbol{w}_{\beta_j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
12:     until maximin stopping criterion
13:     $\sigma_{\text{DEC-grBCM}}^{-2} = M w_{\sigma^{-2},i}^{(\text{end})} + (1 - M w_{\beta_i}^{(\text{end})}) \sigma_{\mathrm{c}}^{-2}$ (13)
14:     $\mu_{\text{DEC-grBCM}} = \sigma_{\text{DEC-grBCM}}^2 (M w_{\mu,i}^{(\text{end})} - (M w_{\beta_i}^{(\text{end})} - 1) \sigma_{\mathrm{c}}^{-2} \mu_{\mathrm{c}})$ (12)
15: end for
```

is multiplied by the number of nodes M and follow (10), (11) to recover the DEC-rBCM mean and precision (Algorithm 4). The local time and space complexity are the same to the rBCM (Table 2). Let $s_{\text{DAC}}^{\text{end}}$ be the maximum number of iterations of three DAC to converge. The total communications are $\mathcal{O}(3 s_{\text{DAC}}^{\text{end}} \text{card}(\mathcal{N}_i))$ (Table 3).

We propose the decentralized grBCM (DEC-grBCM) method which employs three DAC algorithms (Figure 2-(b)) to compute the averages $(1/M)\sum_{i=1}^{M} \beta_i \sigma_{+i}^{-2}$, $(1/M)\sum_{i=1}^{M} \beta_i \sigma_{+i}^{-2} \mu_i$, and $(1/M)\sum_{i=1}^{M} \beta_i$, where $\beta_i = 0.5[\log \sigma_{\mathrm{c}}^2 - \log \sigma_{+i}^2]$. At every iteration of DAC each agent communicates $w_{\mu,i}^{(s)}$, $w_{\sigma^{-2},i}^{(s)}$, $w_{\beta_i}^{(s)}$ to its neighbors $\mathcal{N}_i$. After convergence, each DAC average is multiplied by the number of nodes M and follow (12), (13) to recover the mean and precision (Algorithm 5). The local time and space complexity are the same to the grBCM (Table 2). Let $s_{\text{DAC}}^{\text{end}}$ be the maximum number of iterations of the three DAC to converge. The total communications are $\mathcal{O}(3 s_{\text{DAC}}^{\text{end}} \text{card}(\mathcal{N}_i))$ (Table 3).

Proposition 6. *Let the Assumption 1, 5, 3, 4, 6 hold. If $\omega < 2/M$ then the DEC-grBCM is consistent for any initialization.*

Proof:
Since grBCM is consistent (Proposition 4) and DAC converges for any initial condition (Lemma 1), then DEC-grBCM is consistent for any initialization.

D. Proposed Decentralized Nested Pointwise Aggregation

An additional assumption is required to implement the decentralized NPAE methods.

Algorithm 6 DEC-NPAE [58]

```
Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\boldsymbol{X}$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, $k$, $M$, $\boldsymbol{x}_*$, $\Delta$
Output: $\mu_{\text{DEC-NPAE}}$, $\sigma_{\text{DEC-NPAE}}^2$
 1: initialize $\omega = 2/M$; $\epsilon = 1/\Delta$
 2: for each $i \in \mathcal{V}$ do
 3:     communicate $\boldsymbol{C}_{\theta,i}^{-1}$, $\boldsymbol{X}_i$ to agents in $\mathcal{V} \backslash i$
 4:     $\mu_i \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}_{\theta,i}^{-1})$ (6)
 5:     $[\boldsymbol{k}_A]_i \leftarrow \texttt{crossCov}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \boldsymbol{X}_i, \boldsymbol{C}_{\theta,i}^{-1})$ (14)
 6:     $\text{row}_i\{\boldsymbol{C}_{\theta,A}\} \leftarrow \texttt{localCov}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \boldsymbol{X}, \boldsymbol{C}_{\theta,i}^{-1}, \{\boldsymbol{C}_{\theta,j}^{-1}\}_{j \neq i})$ (15)
 7:     $[\boldsymbol{H}]_i = \text{row}_i\{\boldsymbol{C}_{\theta,A}\}$; $b_{\mu,i} = \mu_i$; $b_{\sigma^2,i} = [\boldsymbol{k}_A]_i$
 8:     initialize $q_{\mu,i}^{(0)} = b_{\mu,i}/[\boldsymbol{H}]_{ii}$, $q_{\sigma^2,i}^{(0)} = b_{\sigma^2,i}/[\boldsymbol{H}]_{ii}$
 9:     repeat                                                    ▷ 2×JOR
10:         communicate $q_{\mu,i}^{(s)}$, $q_{\sigma^2,i}^{(s)}$ to agents in $\mathcal{V} \backslash i$
11:         $q_{\mu,i}^{(s+1)} \leftarrow \texttt{JOR}(\omega, [\boldsymbol{H}]_i, b_{\mu,i}, q_{\mu,i}^{(s)}, \{\boldsymbol{q}_{\mu,j}^{(s)}\}_{j \neq i})$ (19)
12:         $q_{\sigma^2,i}^{(s+1)} \leftarrow \texttt{JOR}(\omega, [\boldsymbol{H}]_i, b_{\sigma^2,i}, q_{\sigma^2,i}^{(s)}, \{\boldsymbol{q}_{\sigma^2,j}^{(s)}\}_{j \neq i})$ (19)
13:     until maximin stopping criterion
14:     initialize $w_{\mu,i}^{(0)} = [\boldsymbol{k}_A]_i q_{\mu,i}^{(\text{end})}$, $w_{\sigma^2,i}^{(0)} = [\boldsymbol{k}_A]_i q_{\sigma^2,i}^{(\text{end})}$
15:     repeat                                                    ▷ 2×DAC
16:         communicate $w_{\mu,i}^{(s)}$, $w_{\sigma^2,i}^{(s)}$ to agents in $\mathcal{N}_i$
17:         $w_{\mu,i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\mu,i}^{(s)}, \{\boldsymbol{w}_{\mu,j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
18:         $w_{\sigma^2,i}^{(s+1)} \leftarrow \texttt{DAC}(\epsilon, w_{\sigma^2,i}^{(s)}, \{\boldsymbol{w}_{\sigma^2,j}^{(s)}\}_{j \in \mathcal{N}_i}, \mathcal{N}_i)$ (18)
19:     until maximin stopping criterion
20:     $\mu_{\text{DEC-NPAE}} = M w_{\mu,i}^{(\text{end})}$
21:     $\sigma_{\text{DEC-NPAE}}^2 = \sigma_f^2 (k_{**} - M w_{\sigma^2,i}^{(\text{end})})$
22: end for
```

Assumption 7. *The graph topology is complete, i.e. every agent i can communicate with every other node $j \neq i$.*

Assumption 7 is conservative, but mandatory for the implementation of the PM and JOR algorithms. In order to use the DEC-NPAE with non-complete but connected graph topologies, flooding is required (Remark 1).

We present DEC-NPAE [58] which combines JOR and DAC to decentralize the computations (16), (17) of NPAE (Figure 3-(a)). We execute two parallel JOR algorithms with known matrix $\boldsymbol{H} = \boldsymbol{C}_{\theta,A}$ and known vectors: i) $\boldsymbol{b} = \boldsymbol{\mu}$; and ii) $\boldsymbol{b} = \boldsymbol{k}_A$. The first JOR is associated with the prediction mean (16) and the second with the variance (17). Implementation details are provided in Algorithm 6. We split up the computation in two parts. First, each entity computes three quantities: i) the local mean μ_i (6); ii) the local cross covariance $[\boldsymbol{k}_A]_i$ (14); and iii) the local row covariance $\text{row}_i\{\boldsymbol{C}_{\theta,A}\}$ (15). For the local computation of (15) the agents must know the inputs $\{\boldsymbol{X}_j\}_{j \neq i}$ of all other agents, to find $\boldsymbol{C}_{\theta,ij}$ and $\boldsymbol{k}_{j,*}$. The inputs $\{\boldsymbol{X}_j\}_{j \neq i}$ are communicated between agents. The local inverted covariance matrices of all other agents $\{\boldsymbol{C}_{\theta,j}^{-1}\}_{j \neq i}$ can be locally computed, but it is computationally expensive to invert $M-1$ matrices, i.e. $\mathcal{O}(M N_i^3) = \mathcal{O}(N^3/M^2)$. Since every agent i has already stored its local covariance matrix from the training step [63], we select to exchange $\{\boldsymbol{C}_{\theta,j}^{-1}\}_{j \neq i}$ between agents (Algorithm 6-[Line 3]). After every JOR iteration, each agent i communicates the computed values

Algorithm 7 PowerMethod

```
Input: $\boldsymbol{R}$, $\mathcal{N}_i$, $M$, $\eta_{\text{PM}}$
Output: $\overline{\lambda}(\boldsymbol{R})$
 1: initialize $\boldsymbol{e}^{(0)} = 1/M$
 2: repeat
 3:     $g_i^{(s+1)} = \text{row}_i\{\boldsymbol{R}\}\boldsymbol{e}^{(s)}$ (20a)
 4:     communicate $g_i^{(s+1)}$ to agents in $\mathcal{V}\backslash i$
 5:     $\|\boldsymbol{g}^{(s+1)}\|_\infty = \max\{|\boldsymbol{g}^{s+1}|\}$
 6:     $\boldsymbol{e}^{(s+1)} = \boldsymbol{g}^{(s+1)}/\|\boldsymbol{g}^{(s+1)}\|_\infty$ (20b)
 7: until $\|\boldsymbol{e}^{(s+1)} - \boldsymbol{e}^{(s)}\|_2 < \eta_{\text{PM}}$
 8: $\overline{\lambda}(\boldsymbol{R}) = \|\boldsymbol{g}^{(\text{end})}\|_\infty$
```

Algorithm 8 DEC-NPAE*

```
Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\boldsymbol{X}$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, $k$, $M$, $\boldsymbol{x}_*$, $\Delta$, $\eta_{\text{PM}}$
Output: $\mu_{\text{DEC-NPAE}^\star}$, $\sigma^2_{\text{DEC-NPAE}^\star}$
 1: for each $i \in \mathcal{V}$ do
 2:     communicate $\text{row}_i\{\boldsymbol{C}_{\theta,A}\}$ to agents in $\mathcal{V}\backslash i$
 3:     $\text{diag}(\boldsymbol{C}_{\theta,A})^{-1} = \text{diag}(\{\boldsymbol{C}_{\theta,A}\}_{ii}^{-1})$
 4:     $\boldsymbol{R} = \text{diag}(\boldsymbol{C}_{\theta,A})^{-1}\boldsymbol{C}_{\theta,A}$
 5:     $\overline{\lambda}(\boldsymbol{R}) \leftarrow \texttt{PowerMethod}(\boldsymbol{R}, \mathcal{N}_i, M, \eta_{\text{PM}})$    ▷ PM1
 6:     $\boldsymbol{B} = \boldsymbol{R} - \overline{\lambda}(\boldsymbol{R}) I_M$
 7:     $\overline{\lambda}(\boldsymbol{B}) \leftarrow \texttt{PowerMethod}(\boldsymbol{B}, \mathcal{N}_i, M, \eta_{\text{PM}})$    ▷ PM2
 8:     $\underline{\lambda}(\boldsymbol{R}) = |\overline{\lambda}(\boldsymbol{B}) - \overline{\lambda}(\boldsymbol{R})|$
 9:     $\omega^\star = 2/(\overline{\lambda}(\boldsymbol{R}) + \underline{\lambda}(\boldsymbol{R}))$
10: end for
11: $\texttt{DEC-NPAE}(\mathcal{D}_i, \boldsymbol{X}, \hat{\boldsymbol{\theta}}, \boldsymbol{C}_{\theta,i}^{-1}, \mathcal{N}_i, k, M, \boldsymbol{x}_*, \Delta, \omega^\star)$
```

$q_{\mu,i}^{(s)}$, $q_{\sigma^2,i}^{(s)}$ to its neighbors $\mathcal{N}_i$ (Algorithm 6-[line 10]). Next, we compute an element of the unknown vectors $q_{\mu,i} = [\boldsymbol{C}_{\theta,A}^{-1}\boldsymbol{\mu}]_i$, $q_{\sigma^2,i} = [\boldsymbol{C}_{\theta,A}^{-1}\boldsymbol{k}_A]_i$ (Algorithm 6-[lines 11, 12]) with the JOR method. When JOR converges, each agent computes locally the i-th element of the resulting summation from: i) the multiplication between the vectors $\boldsymbol{k}_A^\intercal$ and $\boldsymbol{C}_{\theta,A}^{-1}\boldsymbol{\mu}$ (16), that is $w_{\mu,i} = [\boldsymbol{k}_A]_i q_{\mu,i}^{(\text{end})}$; and ii) the multiplication between the vectors $\boldsymbol{k}_A^\intercal$ and $\boldsymbol{C}_{\theta,A}^{-1}\boldsymbol{k}_A$ (17), that is $w_{\sigma^2,i} = [\boldsymbol{k}_A]_i q_{\sigma^2,i}^{(\text{end})}$. Second, since all agents have stored a part of the summations $w_{\mu,i}$, $w_{\sigma^2,i}$, we use the DAC to compute the averages $(1/M)\sum_{i=1}^M [\boldsymbol{k}_A]_i q_{\mu,i}^{(\text{end})}$ and $(1/M)\sum_{i=1}^M [\boldsymbol{k}_A]_i q_{\sigma^2,i}^{(\text{end})}$. After every DAC iteration, each agent i communicates the computed values $w_{\mu,i}^{(s)}$, $w_{\sigma^2,i}^{(s)}$ to its neighbors $\mathcal{N}_i$. When both DAC converge, each agent follows (16), (17) to recover the DEC-NPAE mean and variance. The local time and space complexity are the same to the local NPAE (Table 2). Let $s_{\text{JOR}}^{\text{end}}$ and $s_{\text{DAC}}^{\text{end}}$ be the maximum number of iterations of the JOR and DAC to converge respectively. The total communications for a complete topology yields $\mathcal{O}(2s_{\text{JOR}}^{\text{end}}M + 2s_{\text{DAC}}^{\text{end}}\text{card}(\mathcal{N}_i) + MN_i^2 + MDN_i) = \mathcal{O}(2s_{\text{JOR}}^{\text{end}}M + 2s_{\text{DAC}}^{\text{end}}\text{card}(\mathcal{N}_i) + M(N^2/M^2 + DN/M)$ for all $i \in \mathcal{V}$ as listed in Table 3.

The decentralized NPAE* (DEC-NPAE*) method (Figure 3-(b)) is similar to the DEC-NPAE, but includes an additional routine (Algorithm 7) to compute the optimal relax-

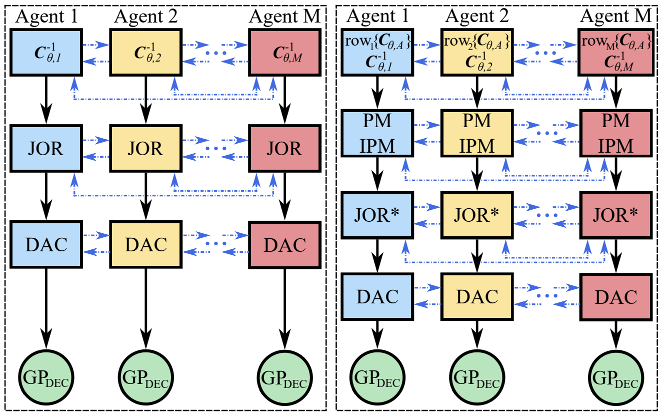

FIGURE 3. **The structure of the DEC-NPAE family. Blue dotted lines correspond to communication (connected and complete). (a) DEC-NPAE [58] incorporates Jacobi over-relaxation (JOR) and discrete-time average consensus (DAC). (b) DEC-NPAE* makes use of the power method (PM) to obtain the optimal relaxation factor and execute JOR*, and DAC.**

ation factor $\omega^\star$ (Lemma 3). More specifically, we employ the PM iterative scheme (20) to estimate the largest $\overline{\lambda}$ and smallest $\underline{\lambda}$ eigenvalues of $\boldsymbol{R}$. To compute the matrix of interest $\boldsymbol{R} = \text{diag}(\boldsymbol{C}_{\theta,A})^{-1}\boldsymbol{C}_{\theta,A}$, each agent i constructs $\boldsymbol{C}_{\theta,A}$ after exchanging $\{\text{row}_j\{\boldsymbol{C}_{\theta,A}\}\}_{j\neq i}$ (Algorithm 8-[Line 2]). Next, each agent i executes the PM (Algorithm 7) to obtain the maximum eigenvalue $\overline{\lambda}(\boldsymbol{R})$. Then, the spectral shift matrix $\boldsymbol{B}$ is composed (Algorithm 8-[line 6]). Using $\boldsymbol{B}$ as an input to the PM algorithm, its maximum eigenvalue is obtained $\overline{\lambda}(\boldsymbol{B})$. To this end, the minimum eigenvalue of $\boldsymbol{R}$ can be computed (Algorithm 7-[Line 8]). Subsequently, the optimal relaxation $\omega^\star$ is computed according to Lemma 3. Provided $\omega^\star$, the DEC-NPAE (Algorithm 6) is executed. Let $s_{\text{PM}}^{\text{end}}$ be the iterations required for the PM to converge. Then, the total communications are $\mathcal{O}(2s_{\text{PM}}^{\text{end}}M + M^2) + \mathcal{O}(\text{DEC-NPAE})$ to exchange: i) the $g_i^{(s)}$ for two PM routines (Algorithm 7-[Line 4]); ii) the $\text{row}_i\{\boldsymbol{C}_{\theta,A}\}$ (Algorithm 8-[Line 2]); and iii) the quantities of DEC-NPAE. The communication complexity for all proposed decentralized methods is shown in Table 3. We illustrate the structure of the DEC-NPAE family in Figure 3.

Proposition 7. *Let Assumption 5, 3, 6, and 7 hold. If $\omega < 2/M$, $\epsilon \in (0, 1/\Delta)$, then the DEC-NPAE is consistent for any initialization. Provided that the conditions for JOR hold for the PM iterations and that $\omega^\star = 2/(\overline{\lambda}(\boldsymbol{R}) + \underline{\lambda}(\boldsymbol{R}))$, then the DEC-NPAE* is consistent for any initial conditions.*

Proof:
Since NPAE is consistent (Proposition 5), and DAC and JOR converge for any initial condition (Lemma 1, 2 respectively), then the DEC-NPAE is consistent for any initialization. The consistency proof for DEC-NPAE* follows similar logic according to Proposition 5 and Lemma 1, 3.

V. Proposed Covariance-Based Nearest Neighbors

In this section, we discuss a distributed algorithm for systems of linear equations (DALE) [7], [59] and propose a new method to identify statistically correlated agents with covariance similarity. The proposed covariance-based nearest neighbor (CBNN) is employed to approximate six aggregation of GP experts methods in a decentralized fashion.

A. Foundational Decentralized Method

An alternative method to solve a linear system of algebraic equations, but for connected (Assumption 1) and not complete topology (Assumption 7) is DALE. This is an iterative method with identical setup to JOR (Section 1) $\boldsymbol{H}\boldsymbol{q} = \boldsymbol{b}$, where $\boldsymbol{H}$ is a known matrix, $\boldsymbol{b}$ a known vector, and $\boldsymbol{q}$ an unknown vector. The i-th node knows: i) i-th row of $\boldsymbol{H}_i = \mathrm{row}_i\{\boldsymbol{H}\} \in \mathbb{R}^{1\times M}$; and ii) i-th entry of $b_i \in \mathbb{R}$. In addition, DALE is formulated as a consensus problem, where the goal for all agents is to obtain the same solution $\boldsymbol{q}_i \in \mathbb{R}^M$ and not just an element of the unknown vector as in JOR. The DALE follows,

$$\boldsymbol{q}_i^{(s+1)} = \boldsymbol{H}_i^{\intercal}(\boldsymbol{H}_i\boldsymbol{H}_i^{\intercal})^{-1}b_i + \frac{1}{\mathrm{card}(\mathcal{N}_i)}\boldsymbol{P}_i \sum_{j\in\mathcal{N}_i} \boldsymbol{q}_j^{(s)}, \quad (21)$$

where $\boldsymbol{P}_i = I_M - \boldsymbol{H}_i^{\intercal}(\boldsymbol{H}_i\boldsymbol{H}_i^{\intercal})^{-1}\boldsymbol{H}_i \in \mathbb{R}^{M\times M}$ is the orthogonal projection onto the kernel of $\boldsymbol{H}_i$.

Assumption 8. *Matrix $\boldsymbol{H}$ is full row rank.*

Lemma 5. *[59, Theorem 3] Let Assumption 1, 8 hold. There exists a constant $\phi \in (0,1)$ such that all $\boldsymbol{q}_i^{(s)}$ converge to the solution for any initialization $\boldsymbol{q}_i^{(0)}$ with worst case convergence speed ϕ^s.*

The convergence speed constant ϕ depends on the number of agents M and the diameter $\mathrm{diam}(\mathcal{G})$. The larger the fleet size and the diameter the slower the convergence. The i-th node using DALE (21) exchanges information only with its neighbors $j \in \mathcal{N}_i$ and not with the whole network (see in contrast Remark 1 for JOR). In addition, DALE is concurrently a consensus algorithm and updates the vector $\boldsymbol{q}_i^{(s)} \in \mathbb{R}^M$, while JOR updates just the corresponding entry $[\boldsymbol{q}_i^{(s)}]_i \in \mathbb{R}$. Thus, DALE produces similar results to the sequential operation of JOR and DAC.

B. Covariance-Based Nearest Neighbors (CBNN)

To identify statistically correlated agents for a location of interest $\boldsymbol{x}_*$ we introduce the covariance-based nearest neighbor (CBNN) method. Let every agent i to have its own opinion for the location of interest $\{\mu_1,\ldots,\mu_M\}$, where $\mu_i = \mathrm{E}[y(\boldsymbol{x}_*) \mid \mathcal{D}_i, \boldsymbol{\theta}]$ computed as a GP local mean (6). We employ the local mean values to form the *mean dataset* $\mathcal{D}_\mu = (\{\boldsymbol{X}_i\}_{i=1}^M, \{\mu_i\}_{i=1}^M) = (\boldsymbol{X}, \boldsymbol{\mu})$, where $\boldsymbol{X}_i \in \mathbb{R}^{D\times N_i}$, $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, $\mu_i \in \mathbb{R}$, and $\boldsymbol{\mu} \in \mathbb{R}^M$.

Definition 2. *Let the vector of random variables $(\mu_1(\boldsymbol{x}_*),\ldots,\mu_M(\boldsymbol{x}_*), y(\boldsymbol{x}_*))^{\intercal} \in \mathbb{R}^{M+1}$ to form a random process, where the first two moments exist with zero mean $\mu_\mu = 0$ and a finite covariance $\boldsymbol{K}_{\theta,\mu}$.*

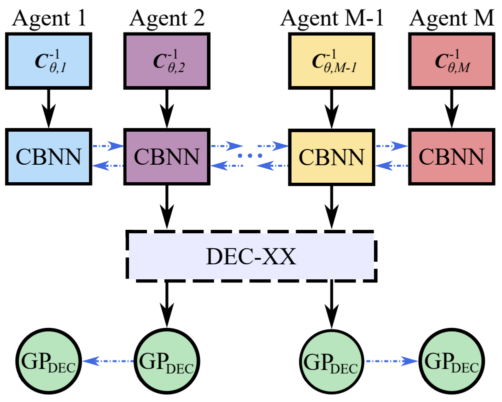

FIGURE 4. The structure of the proposed nearest neighbor decentralized aggregation methods. Blue dotted lines correspond to communication (connected). The covariance-based nearest neighbor (CBNN) method identifies statistically correlated agents—in this illustration the CBNN set is $\mathcal{V}_{\mathrm{NN}} \in [2, M-1]$. Next, a decentralized aggregation method among the DEC-PoE and DEC-BCM families is executed within the $\mathcal{V}_{\mathrm{NN}}$ nodes. After convergence, the predicted values are communicated to the rest agents of the network.

Proposition 8. *[Appendix B] The random process (Definition 2) approximates a Gaussian process, $(\mu_1(\boldsymbol{x}_*),\ldots,\mu_M(\boldsymbol{x}_*), y(\boldsymbol{x}_*))^{\intercal} \sim \mathcal{GP}(\boldsymbol{\mu}_\mu, \boldsymbol{K}_{\theta,\mu})$ as $N \to \infty$.*

The covariance of the new GP (Proposition 8) yields,

$$\boldsymbol{C}_{\theta,\mu} = \mathrm{Cov}[\boldsymbol{\mu}(\boldsymbol{x}_*), y(\boldsymbol{x}_*)] = \begin{bmatrix} \boldsymbol{K}_\mu & \boldsymbol{k}_{\mu,*}^{\intercal} \\ \boldsymbol{k}_{\mu,*} & k_{**} \end{bmatrix},$$

where $\boldsymbol{k}_{\mu,*}^{\intercal} \in \mathbb{R}^M$ is the cross-covariance. Interestingly, the cross-covariance element of the i-th agent represents the correlation of a local dataset $\mathcal{D}_i$ to the location of interest $\boldsymbol{x}_*$ with a positive scalar number $[\boldsymbol{k}_{\mu,*}]_i \in \mathbb{R}_{\geq 0}$. The physical interpretation of the cross-covariance is that when this value approaches zero $[\boldsymbol{k}_{\mu,*}]_i \to 0$, the agent i becomes statistically uncorrelated to the location of interest $\boldsymbol{x}_*$. Notably, each agent i can compute locally its cross-covariance element without any communication with neighboring agents,

$$[\boldsymbol{k}_{\mu,*}]_i = \boldsymbol{k}_{i,*}^{\intercal}\boldsymbol{C}_{\theta,i}^{-1}\boldsymbol{k}_{i,*}, \quad (22)$$

where $\boldsymbol{k}_{i,*} = k(\boldsymbol{X}_i, \boldsymbol{x}_*)$. As a result, every agent can locally determine a non-negative similarity score for the location of interest. If the similarity score is very low, then the agent is excluded from the aggregation. The workflow proceeds as follows. Every agent i computes its cross-covariance $[\boldsymbol{k}_{\mu,*}]_i$ (22). When the correlation of agent i to the location of interest is below a threshold $[\boldsymbol{k}_{\mu,*}]_i < \eta_{\mathrm{NN}}$, then the agent is excluded from the aggregation of GP experts. After all agents compute their correlations, the nearest neighbor subset of nodes is determined $\mathcal{V}_{\mathrm{NN}} \subseteq \mathcal{V}$ with $M_{\mathrm{NN}} = \mathrm{card}(\mathcal{V}_{\mathrm{NN}}) \leq M$. The excluded agents are not contributing on $y(\boldsymbol{x}_*)$ and

Algorithm 9 DEC-NN-PoE

Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, k, M, $\boldsymbol{x}_*$, Δ, η_{NN}

Output: $\mu_{\text{DEC-NN-PoE}}$, $\sigma_{\text{DEC-NN-PoE}}^{-2}$

```
1:  for each $i \in \mathcal{V}$ do
2:      $[\boldsymbol{k}_{\mu,*}]_i \leftarrow \texttt{CrossCovCBNN}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \boldsymbol{X}_i, \boldsymbol{C}_{\theta,i}^{-1})$ (22)
3:      for each $j \in \mathcal{N}_i$ do
4:          if $[\boldsymbol{k}_{\mu,*}]_j < \eta_{\text{NN}}$ then
5:              $\mathcal{N}_{\text{NN},i} = \mathcal{N}_i \backslash j$
6:              $j \leftarrow \texttt{flooding}(\mathcal{V})$
7:              $\mathcal{V}_{\text{NN}} = \mathcal{V} \backslash j$
8:          end if
9:      end for
10:     $M_{\text{NN}} = \text{card}(\mathcal{V}_{\text{NN}})$
11: end for
12: $\texttt{DEC-PoE}(\mathcal{D}_i, \hat{\boldsymbol{\theta}}, \boldsymbol{C}_{\theta,i}^{-1}, \mathcal{N}_{\text{NN},i}, k, M_{\text{NN}}, \boldsymbol{x}_*, \Delta)$
13: communicate $\mu_{\text{DEC-NN-PoE}}$ and $\sigma^2_{\text{DEC-NN-PoE}}$ to agents in $\mathcal{V} \backslash \mathcal{V}_{\text{NN}}$
```

Algorithm 10 DEC-NN-gPoE

Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, k, M, $\boldsymbol{x}_*$, Δ, η_{NN}

Output: $\mu_{\text{DEC-gPoE}}$, $\sigma_{\text{DEC-gPoE}}^{-2}$

```
1: Same to Algorithm 9 with DEC-gPoE instead of DEC-PoE
```

wait to receive the predicted values after the aggregation is complete.

Lemma 6. *[Appendix C] Let the agents operate in a spatial environment with input space of dimension $D = 2$. Each agent i collects local data $\mathcal{D}_i$ from a disjoint partition in stripes along the y-axis (Figure 6-(b)) and the network of agents form a line graph topology. Then, CBNN preserves network connectivity.*

The advantages of using CBNN to identify statistically correlated agents are: i) the selection of nearest neighbors is justified through a covariance not just by using an arbitrary radius; ii) only the local dataset $\mathcal{D}_i$ is required to compute (22) with no data exchange; iii) the total communications are reduced, as a subset of the agents takes part to the aggregation $\mathcal{V}_{\text{NN}}$; iv) the DAC converges faster (Lemma 1); and v) the DALE can be employed as $\boldsymbol{H}$ is ensured to be full row rank.

C. Proposed Decentralized Nearest Neighbors

1) DEC-NN-PoE Family

The decentralized nearest neighbor PoE (DEC-NN-PoE) family integrates CBNN to the DEC-PoE family as shown in Figure 4. The implementation details for DEC-NN-PoE are given in Algorithm 9 and for DEC-NN-gPoE in Algorithm 10. Every agent i computes the local cross-covariance of CBNN $[\boldsymbol{k}_{\mu,*}]_i$ (22) and evaluates its involvement to the aggregation (Algorithm 9-[Line 4]). After the CBNN terminates, the remaining agents $\mathcal{V}_{\text{NN}}$ run the DEC-PoE family routines (Algorithm 1, 2). Finally, the predicted values are transmitted to the excluded agents $\mathcal{V}\backslash\mathcal{V}_{\text{NN}}$. The communication complexity for both methods is $\mathcal{O}(2s_{\text{DAC}}^{\text{end}}\text{card}(\mathcal{N}_{\text{NN},i}))$.

Algorithm 11 DEC-NN-BCM

Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, k, M, $\boldsymbol{x}_*$, Δ, η_{NN}

Output: $\mu_{\text{DEC-BCM}}$, $\sigma_{\text{DEC-BCM}}^{-2}$

```
1: Same to Algorithm 9 with DEC-BCM instead of DEC-PoE
```

Algorithm 12 DEC-NN-rBCM

Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,i}^{-1}$, $\mathcal{N}_i$, k, M, $\boldsymbol{x}_*$, Δ, η_{NN}

Output: $\mu_{\text{DEC-rBCM}}$, $\sigma_{\text{DEC-rBCM}}^{-2}$

```
1: Same to Algorithm 9 with DEC-rBCM instead of DEC-PoE
```

Algorithm 13 DEC-NN-grBCM

Input: $\mathcal{D}_{+i}(\boldsymbol{X}_{+i}, \boldsymbol{y}_{+i})$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}_{\theta,+i}^{-1}$, $\mathcal{N}_i$, k, M, $\boldsymbol{x}_*$, Δ, η_{NN}

Output: $\mu_{\text{DEC-NN-grBCM}}$, $\sigma_{\text{DEC-NN-grBCM}}^{-2}$

```
1: Same to Algorithm 9 with DEC-grBCM instead of DEC-PoE
```

2) DEC-NN-BCM Family

The decentralized nearest neighbor BCM (DEC-NN-BCM) family is the DEC-BCM family with a CBNN selection (Figure 4). The implementation details for DEC-NN-BCM are given in Algorithm 11, for DEC-NN-rBCM in Algorithm 12, and for DEC-NN-grBCM in Algorithm 13. The communication complexity for DEC-NN-BCM and DEC-NN-rBCM is $\mathcal{O}(2s_{\text{DAC}}^{\text{end}}\text{card}(\mathcal{N}_{\text{NN},i}))$, while for DEC-NN-grBCM is $\mathcal{O}(3s_{\text{DAC}}^{\text{end}}\text{card}(\mathcal{N}_{\text{NN},i}))$.

Proposition 9. *Let Assumption 1, 5, 3, 4, 6 hold. If $\omega < 2/M$ then the DEC-NN-grBCM is consistent for any initialization.*

Proof:
In Proposition 6, we show that DEC-grBCM is consistent. Since we proved that CBNN preserves network connectivity (Lemma 6), then DEC-NN-grBCM is consistent for any initialization.

3) DEC-NN-NPAE

We introduce the decentralized nearest neighbor NPAE (DEC-NN-NPAE) method to distribute the computations (16), (17) of NPAE (Figure 5). The DEC-NN-NPAE employs the CBNN and DALE (21) methods. By using CBNN, one can satisfy Assumption 8 and use the DALE. Thus, the DEC-NN-NPAE relaxes the complete topology (Assumption 7) to a connected topology. Implementation details are provided in Algorithm 14. First, each entity computes: i) the local cross covariance $[\boldsymbol{k}_A]_i$ (14); and ii) the cross-covariance of CBNN $[\boldsymbol{k}_{\mu,*}]_i$ (22). Next, we execute the CBNN routine to select the nearest neighbors. During the CBNN, if a criterion is met for an agent j to be excluded

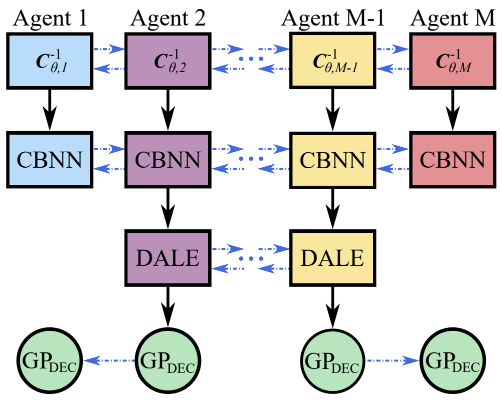

FIGURE 5. **The structure of the proposed nearest neighbor decentralized aggregation methods. Blue dotted lines correspond to communication (connected). The covariance-based nearest neighbor (CBNN) method identifies statistically correlated agents—in this illustration the CBNN set is $\mathcal{V}_{\text{NN}} \in [2, M-1]$. Next, a distributed algorithm for solving a linear system of equations (DALE) is executed within the $\mathcal{V}_{\text{NN}}$ nodes. After convergence, the predicted values are communicated to the excluded agents of the network.**

(Algorithm 14-[Line 5]), it is removed from the list of agents $\mathcal{V}_{\text{NN}} = \mathcal{V}\backslash j$; and if not, the corresponding element of the local cross covariance $[\boldsymbol{k}_A]_j$ is communicated to all other agents $\mathcal{V}_{\text{NN}}\backslash i$. When the CBNN routine terminates, we execute the DALE method on the nearest neighbors $\mathcal{V}_{\text{NN}}$. Similarly to DEC-NPAE, the inputs $\{\boldsymbol{X}_j\}_{j\neq i}$ and the local inverted covariance matrices $\{\boldsymbol{C}^{-1}_{\theta,j}\}_{j\neq i}$ are communicated between CBNN agents. Next, we execute two parallel DALE algorithms with known matrix $\boldsymbol{H} = \boldsymbol{C}_{\theta,A}$ and known vectors: i) $\boldsymbol{b} = \boldsymbol{\mu}$; and ii) $\boldsymbol{b} = \boldsymbol{k}_A$. The first DALE is associated with the prediction mean $\mu_{\text{DEC-NN-NPAE}}$ (Algorithm 14-[Line 24]) and the second with the variance $\sigma^2_{\text{DEC-NN-NPAE}}$ (Algorithm 14-[Line 25]). After every DALE iteration, each agent i communicates the computed vectors $\boldsymbol{q}^{(s)}_{\mu,i}$, $\boldsymbol{q}^{(s)}_{\sigma^2,i}$ to its neighbors $\mathcal{N}_{\text{NN},i}$ (Algorithm 14-[Line 23]). Next, we update the vectors $\boldsymbol{q}_{\mu,i}$, $\boldsymbol{q}_{\sigma^2,i}$ (Algorithm 6-[Lines 24, 25]) with the DALE method. When both DALE converge, each agent follows (16), (17) to recover the DEC-NN-NPAE mean and variance. Let $s^{\text{end}}_{\text{DALE}}$ be the maximum number of iterations of DALE to converge. The total communications during the CBNN yields $\mathcal{O}(M_{\text{NN}})$ and during DALE $\mathcal{O}(2s^{\text{end}}_{\text{DALE}}\text{card}(\mathcal{N}_{\text{NN},i}) + M_{\text{NN}}N_i^2 + M_{\text{NN}}DN_i) = \mathcal{O}(2s^{\text{end}}_{\text{DALE}}\text{card}(\mathcal{N}_{\text{NN},i}) + M_{\text{NN}}(N^2/M^2_{\text{NN}} + DN/M_{\text{NN}}))$.

Proposition 10. *Let Assumption 1, 5, 3, 6 hold. Then, the DEC-NN-NPAE is consistent for any initialization of DALE.*

Proof:
Since we know that NPAE is consistent (Proposition 5), DALE converges for any initial condition (Lemma 5), and CBNN preserves network connectivity (Lemma 6), then DEC-NN-NPAE is consistent for any initialization.

VI. Numerical Experiments

In this section, we conduct numerical experiments to demonstrate the efficacy of the proposed decentralized GP prediction methods. Two real-world datasets of sea surface temperature (SST) [70], [71] and ground elevation map [72], [73] are used to assess the GP prediction algorithms in four aspects: i) prediction accuracy; ii) uncertainty quantification; iii) communications per agent; and iv) execution time. All numerical experiments are conducted in MATLAB using the GPML package [74] on an Intel Core i7-6700 CPU @3.40 GHz with 32.0 GB memory RAM[1].

Algorithm 14 DEC-NN-NPAE [58]

Input: $\mathcal{D}_i(\boldsymbol{X}_i, \boldsymbol{y}_i)$, $\boldsymbol{X}$, $\hat{\boldsymbol{\theta}}$, $\boldsymbol{C}^{-1}_{\theta,i}$, $\mathcal{N}_i$, k, M, $\boldsymbol{x}_*$, Δ, η_{NN}
Output: $\mu_{\text{DEC-NN-NPAE}}$, $\sigma^2_{\text{DEC-NN-NPAE}}$

```
 1: for each $i \in \mathcal{V}$ do
 2:     $[\boldsymbol{k}_A]_i \leftarrow \texttt{crossCov}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \boldsymbol{X}_i, \boldsymbol{C}^{-1}_{\theta,i})$ (14)
 3:     $[\boldsymbol{k}_{\mu,*}]_i \leftarrow \texttt{CrossCovCBNN}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \boldsymbol{X}_i, \boldsymbol{C}^{-1}_{\theta,i})$ (22)
 4:     for each $j \in \mathcal{N}_i$ do
 5:         if $[\boldsymbol{k}_{\mu,*}]_j < \eta_{\text{NN}}$ then
 6:             $\mathcal{N}_{\text{NN},i} = \mathcal{N}_i\backslash j$; $\mathcal{V}_{\text{NN}} = \mathcal{V}\backslash j$
 7:             $j \leftarrow \texttt{flooding}(\mathcal{V}_{\text{NN}})$
 8:         else
 9:             $[\boldsymbol{k}_A]_j \leftarrow \texttt{flooding}(\mathcal{V}_{\text{NN}})$
10:         end if
11:     end for
12: end for
13: for each $i \in \mathcal{V}_{\text{NN}}$ do
14:     $\mu_i \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \hat{\boldsymbol{\theta}}, \mathcal{D}_i, \boldsymbol{C}^{-1}_{\theta,i})$ (6)
15:     $\boldsymbol{C}^{-1}_{\theta,i}, \boldsymbol{X}_i \leftarrow \texttt{flooding}(\mathcal{V}_{\text{NN}})$
16:     $\boldsymbol{k}_{\text{NN},A} = [\boldsymbol{k}_A]_i \cup \{[\boldsymbol{k}_A]_j\}_{j\in\mathcal{V}_{\text{NN}}}$
17:     $\text{row}_{\text{NN},i}\{\boldsymbol{C}_{\theta,A}\} \leftarrow \texttt{localCov}(\boldsymbol{x}_*, k, \boldsymbol{X}, \hat{\boldsymbol{\theta}}, \mathcal{V}_{\text{NN}})$ (15)
18:     $\boldsymbol{H}_i = \text{row}_{\text{NN},i}\{\boldsymbol{C}_{\theta,A}\}$; $M_{\text{NN}} = \text{card}(\mathcal{V}_{\text{NN}})$
19:     $b_{\mu,i} = \mu_{\text{NN},i}$; $\boldsymbol{b}_{\sigma^2} = \boldsymbol{k}_{\text{NN},A}$
20:     $\boldsymbol{P}_i = I_{M_{\text{NN}}} - \boldsymbol{H}_i^\intercal(\boldsymbol{H}_i\boldsymbol{H}_i^\intercal)^{-1}\boldsymbol{H}_i$
21:     initialize $\boldsymbol{q}^{(0)}_{\mu,i} = b_{\mu,i} \oslash \boldsymbol{H}_i$; $\boldsymbol{q}^{(0)}_{\sigma^2,i} = \boldsymbol{b}_{\sigma^2} \oslash \boldsymbol{H}_i$
22:     repeat                                                  ▷ 2×DALE (21)
23:         communicate $\boldsymbol{q}^{(s)}_{\mu,i}$, $\boldsymbol{q}^{(s)}_{\sigma^2,i}$ to neighbors $\mathcal{N}_{\text{NN},i}$
24:         $\boldsymbol{q}^{(s+1)}_{\mu,i} \leftarrow \texttt{DALE}(\boldsymbol{P}_i, \boldsymbol{H}_i, b_{\mu,i}, \{\boldsymbol{q}^{(s)}_{\mu,j}\}_{j\in\mathcal{N}_{\text{NN},i}}, \mathcal{N}_{\text{NN},i})$
25:         $\boldsymbol{q}^{(s+1)}_{\sigma^2,i} \leftarrow \texttt{DALE}(\boldsymbol{P}_i, \boldsymbol{H}_i, b_{\sigma^2,i}, \{\boldsymbol{q}^{(s)}_{\sigma^2,j}\}_{j\in\mathcal{N}_{\text{NN},i}}, \mathcal{N}_{\text{NN},i})$
26:     until maximin stopping criterion
27:     $\mu_{\text{DEC-NN-NPAE}} = \boldsymbol{k}^\intercal_{\text{NN},A}\boldsymbol{q}^{\text{end}}_{\mu,i}$
28:     $\sigma^2_{\text{DEC-NN-NPAE}} = \sigma_f^2(k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}^\intercal_{\text{NN},A}\boldsymbol{q}^{\text{end}}_{\sigma^2,i})$
29: end for
```

A. Datasets and Experimental Design

We evaluate our methods on two real-world datasets: D1) Sea surface temperature (SST) from NASA JPL [70], [71]; and D2) Ground elevation maps from NASA Shuttle Radar Topography Mission [72]. For D1, we extract $123,200$ SST values in Kelvins from the region (36.4^{o} N, -73.0^{o} E) to (40.0^{o} N, -69.4^{o} E) covering approximately $400\,\text{km} \times 400\,\text{km}$ of the Atlantic ocean. For D2, we use $1,563,750$ elevation values from the N43W080 tile, a region with significant elevation difference. Both datasets are normalized

[1]Demonstration code: github.com/gkontoudis/decentralized-GP.

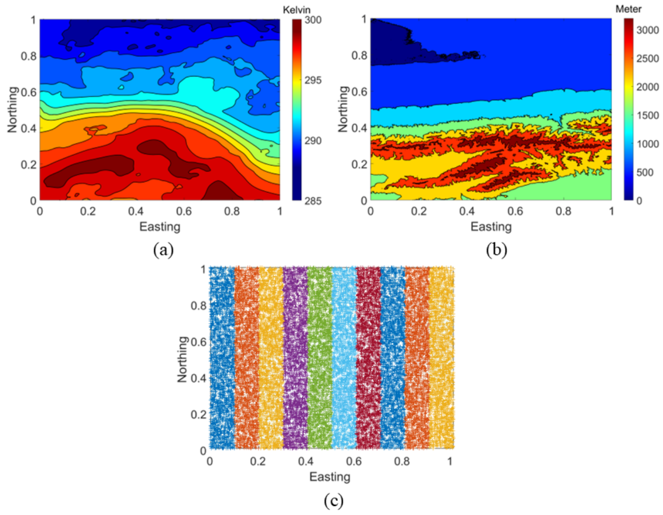

FIGURE 6. **(a) Sea surface temperature field [71]; (b) Ground elevation map [72]; (c) Non-overlapping space partition for $M = 10$ agents.**

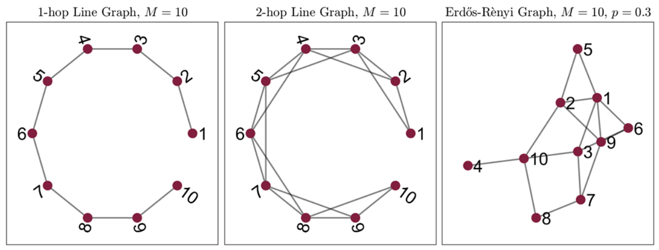
1-hop Line Graph, $M = 10$
2-hop Line Graph, $M = 10$
Erdős-Rènyi Graph, $M = 10$, $p = 0.3$

FIGURE 7. **Three graph topologies are evaluated for 10 agents. The 1-hop line graph represents the most parsimonious topology, as the failure of any edge leads to disconnection. Thus, the line graph serves as a lower bound on both the prediction accuracy and uncertainty quantification.**

over $[0, 1]^2$ as shown in Figure 6(a)-(b). Notably, D2 is characterized as a nonstationary environment [75], making it particularly challenging for predictive performance. We formulate 15 datasets of size $N = 20,000$ to avoid random assignment of data. Each dataset is equally distributed for four fleet sizes $M = \{4, 10, 20, 40\}$, e.g., Figure 6(c) illustrates the case of $M = 10$ agents. This setup represents the worst-case partitioning of the environment, as there is no overlap and agents remain agnostic to global variability. Such a design highlights the role of space partitioning in distributed learning, since stronger connectivity or overlapping partitions can improve the predictive performance.

The GP training of the hyperparameters is performed with the DEC-gapx-GP method [42]. We employ the proposed methods over $N_t = 100$ prediction points: i) DEC-PoE; ii) DEC-NN-PoE; iii) DEC-gPoE; iv) DEC-NN-gPoE; v) DEC-BCM; vi) DEC-NN-BCM; vii) DEC-rBCM; viii) DEC-NN-rBCM; ix) DEC-grBCM; x) DEC-NN-grBCM; xi) DEC-NPAE [58]; xii) DEC-NPAE*; and xiii) DEC-NN-NPAE [58]. All methods are implemented in a one-hop line graph, two-hop line graph, and random graph topology except of (xi) and (xii). An example of one-hop line graph, two-hop line graph, and random graph topology for 10

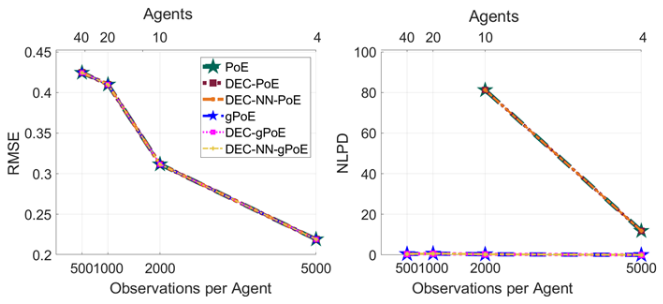

FIGURE 8. **Average RMSE and NLPD values for four fleet sizes with the PoE-based methods on any graph topology.**

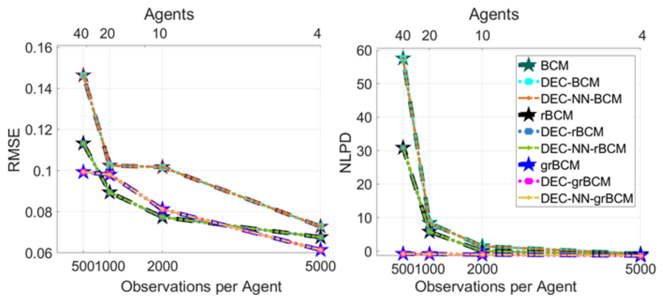

FIGURE 9. **Average RMSE and NLPD values for four fleet sizes with the BCM-based methods on any graph topology.**

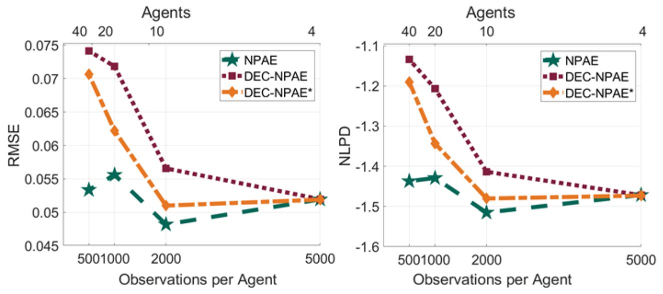

FIGURE 10. **Average RMSE and NLPD values for four fleet sizes with the NPAE-based methods on a any graph topology.**

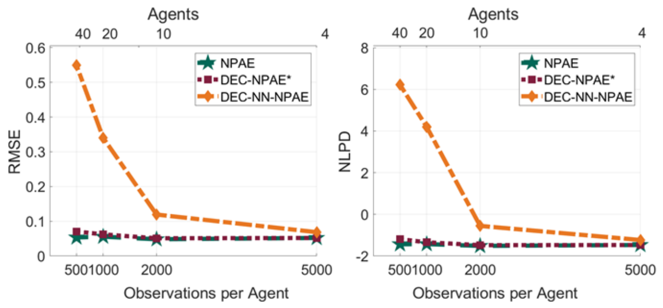

FIGURE 11. **Average RMSE and NLPD values for four fleet sizes with the NPAE-based methods on a complete graph topology.**

agents is shown in Figure 7. The Erdős-Rényi random graphs follow $\mathcal{G}(M = 4, p = 0.6)$, $\mathcal{G}(M = 10, p = 0.3)$, $\mathcal{G}(M = 20, p = 0.2)$, and $\mathcal{G}(M = 40, p = 0.15)$ for 4, 10, 20, and 40 agents, respectively. Line graph is the most parsimonious topology (i.e., single edge removal will cause a network disconnection) and serves as a lower bound for our

TABLE 4. Nearest Neighbors, Execution Time, and Communication Rounds of Decentralized Nearest Neighbor Methods for SST Dataset D1.

			One-hop Line Graph		Two-hop Line Graph		Random Graph	
Number of Agents M	Method	Nearest Neighbors	Average Time per Agent [s]	Average Comms s^{end}	Average Time per Agent [s]	Average Comms s^{end}	Average Time per Agent [s]	Average Comms s^{end}
4	DEC-NN-PoE	2.3±0.1	0.0312	3.6	0.0205	3.4	0.0179	3.4
	DEC-NN-gPoE		0.0339	3.6	0.0332	3.4	0.0335	3.4
	DEC-NN-BCM		0.0323	3.6	0.0317	3.4	0.0318	3.4
	DEC-NN-rBCM		0.0469	3.6	0.0450	3.4	0.0455	3.4
	DEC-NN-grBCM		0.0516	3.6	0.0499	3.4	0.0512	3.4
	DEC-NN-NPAE [58]		1.1683	69.4	1.1592	66.8	1.1637	67.7
10	DEC-NN-PoE	5.7±0.2	0.0122	7.2	0.0112	6.7	0.0109	6.5
	DEC-NN-gPoE		0.0121	7.2	0.0112	6.7	0.0109	6.5
	DEC-NN-BCM		0.0126	7.2	0.0114	6.7	0.0113	6.5
	DEC-NN-rBCM		0.0172	7.2	0.0162	6.7	0.0154	6.5
	DEC-NN-grBCM		0.0167	7.2	0.0159	6.7	0.0152	6.5
	DEC-NN-NPAE [58]		0.4844	250.2	0.4451	247.6	0.4344	227.9
20	DEC-NN-PoE	11.3±0.4	0.0087	6.8	0.0075	5.9	0.0068	5.2
	DEC-NN-gPoE		0.0083	6.8	0.0079	5.9	0.0075	5.2
	DEC-NN-BCM		0.0086	6.8	0.0084	5.9	0.0079	5.2
	DEC-NN-rBCM		0.0126	7.0	0.0118	6.0	0.0100	5.5
	DEC-NN-grBCM		0.0124	7.0	0.0109	6.0	0.0099	5.5
	DEC-NN-NPAE [58]		0.2698	625.6	0.2312	546.3	0.2077	494.9
40	DEC-NN-PoE	23.6±1.1	0.0052	7.4	0.0046	6.7	0.0043	5.8
	DEC-NN-gPoE		0.0049	7.4	0.0043	6.7	0.0039	5.8
	DEC-NN-BCM		0.0051	7.4	0.0045	6.7	0.0042	5.8
	DEC-NN-rBCM		0.0073	7.4	0.0068	6.7	0.0061	5.8
	DEC-NN-grBCM		0.0071	7.4	0.0064	6.7	0.0059	5.8
	DEC-NN-NPAE [58]		2.7009	1,824.1	2.4381	1,651.9	2.2393	1,445.3

algorithms. We anticipate our methods to increase their computation and communication efficiency in topologies with higher connectivity. The NN threshold is set to $\eta_{\text{NN}} = 10^{-3}$.

The quality assessment is accomplished with the root mean square error, RMSE $= [1/N \sum_{i=1}^{N} (\mu(\boldsymbol{x}_*) - y(\boldsymbol{x}_*))^2]^{1/2}$, the normalized root mean square error, NRMSE $=$ RMSE$/(y_{\text{max}} - y_{\text{min}})$ for prediction accuracy, and the negative log predictive density, NLPD $= -1/N \sum_{i=1}^{N} \log p(\hat{\boldsymbol{y}}_* \mid \mathcal{D}, \boldsymbol{x}_*)$ for uncertainty quantification, where $p(\hat{\boldsymbol{y}}_* \mid \mathcal{D}, \boldsymbol{x}_*)$ is the predictive distribution.

B. Convergence, Accuracy & Uncertainty Quantification

In Figure 8, 13, we report the average RMSE, NRMSE, and NLPD values for datasets D1 and D2, respectively. We evaluate four fleet sizes using decentralized PoE-based methods. The RMSE and NRMSE performance is identical for all PoE-based methods (DEC-(NN)-(g)PoE) with their centralized counterparts PoE [31] and gPoE [33], showing that the proposed decentralized methods converge with negligible approximation error. In addition, the same results are consistently reproduced across all three network topologies. Since convergence is achieved under the parsimonious one-hop line graph, convergence is also guaranteed for the other two graph topologies. Consistent with Proposition 2, PoE and gPoE produce identical mean predictions. For uncertainty quantification, both DEC-(NN)-PoE match the NLPD values of PoE. In the SST dataset D1, all PoE-based methods fail to provide finite NLPD values for larger fleet sizes ($M = 20$ and $M = 40$ agents), due to the additive form of the predictive variance in PoE (9), which yields overconfident estimates. By contrast, in the ground elevation dataset D2, uncertainty is successfully quantified across all fleet sizes. Notably, both nearest neighbor methods DEC-NN-(g)PoE perform indistinguishably from their (g)PoE counterparts, despite excluding on average 42.5% of agents for D1 and 23.9% of agents for D2 during the aggregation (Table 4, 5).

In Figure 9, 14, we present the average RMSE, NRMSE, and NLPD values for both D1 and D2 datasets. We deploy four fleet sizes using the decentralized BCM-based methods in three network topologies that report matching performance. The proposed BCM-family methods (DEC-(NN)-(gr)(r)BCM) are tested against the centralized BCM [30]; rBCM [32]; and grBCM [35]. We observe that all proposed decentralized BCM-family methods converge to the BCM optimal values as they report identical RMSE, NRMSE, and NLPD values for both datasets D1 and D2 in Figure 9, 14,

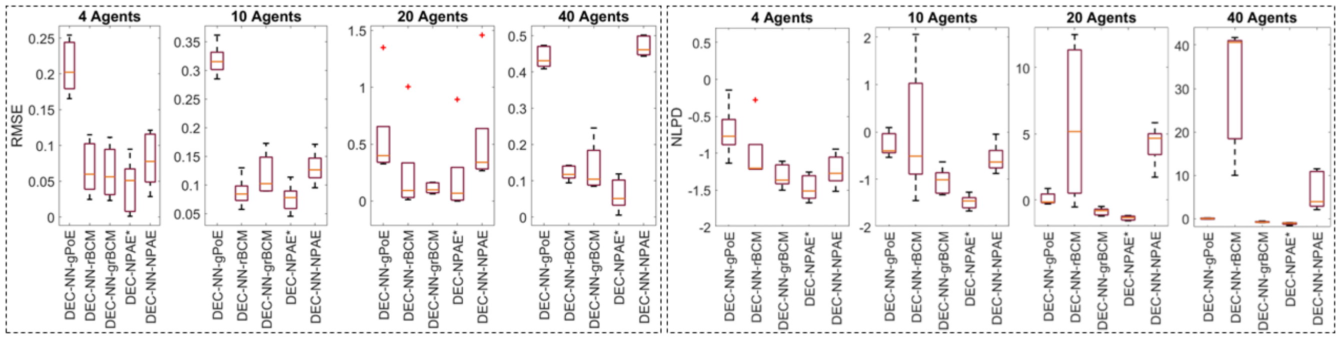

FIGURE 12. Comparison of accuracy and uncertainty quantification for four fleet sizes and 15 replications at $N_t = 100$ unknown locations using $N = 20,000$ observations from the Sea Surface Temperature dataset in a one-hop line graph topology. Lower RMSE and NLPD values indicate better accuracy and better uncertainty quantification respectively. The comparison includes the five best decentralized GP prediction methods out of 13. DEC-NN-NPAE was introduced in our previous work [58].

respectively. Importantly, all nearest neighbor methods DEC-(NN)-(gr)(r)BCM converge to the optimal values, although a subset of agents is selected to participate in the prediction.

The average RMSE, NRMSE, and NLPD values for four fleet sizes are presented in Figure 10, 11 and Figure 15, 16 using the decentralized NPAE-based methods for datasets D1 and D2, respectively. We test four fleet sizes with the decentralized NPAE-based methods. Figure 10, 15 test one-hop, two-hop, and random graph topologies, but the choice of topology does not affect the results. Figure 11, 16 focus on the complete graph topology, which is required from DEC-NPAE* and DEC-NN-NPAE [58]. Since the decentralized NPAE-based methods are designed to approximate the centralized NPAE, the optimal RMSE, NRMSE, and NLPD values are that of NPAE [36]. The key difference between DEC-NPAE [58] and DEC-NPAE* lies in the estimation of the optimal relaxation factor of the latter, as opposed to the heuristic selection based on fleet size in the former (Remark 2). All decentralized NPAE-family methods report an approximation error, and thus do not converge to the optimal values of NPAE. More specifically, DEC-NPAE* yields the smallest approximation error (Figure 10), while DEC-NN-NPAE reports the highest approximation error (Figure 11) for dataset D1. In contrast, for dataset D2, all methods achieve similar prediction accuracy and uncertainty quantification, showing that the optimal relaxation factor does not always improve accuracy. Nonetheless, as discussed in the complexity analysis (Table 3), DEC-NPAE* converges faster than DEC-NPAE, which advocates that the proposed method with optimal relaxation factor, is always more efficient in speed with comparable or better prediction accuracy.

The convergence of the proposed decentralized methods to their centralized counterparts with an one-hop line graph topology implies that any graph topology will convergence with similar or better convergence speed. This is validated with our experiments on a two-hop line graph and random graph topology that produce identical prediction accuracy (RMSE, NRMSE) and uncertainty quantification (NLPD) results for all methods, fleet sizes, and datasets.

C. Nearest Neighbors, Execution Time & Communication

In Table 4 and 5, we report the average wall-clock execution time per agent and the number of communication rounds required for convergence across all nearest neighbor methods, tested on a one-hop line, two-hop line, and random graph topologies for datasets D1 and D2, respectively. In addition, we compute the average number of nearest neighbors M_{NN} of each family that participates in the prediction for all $N_t = 100$ prediction points of each fleet size.

For D1, the results show a 42.5% agent reduction in participating agents with no approximation error for DEC-NN-(g)PoE and DEC-NN-(gr)(r)BCM (Figure 8, 9), while DEC-NN-NPAE [58] reports high approximation error for (Figure 11). In addition, increasing network connectivity consistently reduces both execution time and communication rounds across all methods (Table 4). Note that DEC-NN-NPAE [58] remains computationally intensive and requires substantially more communication rounds to converge. For D2, the agent reduction is on average 23.9% with no approximation error for DEC-NN-(g)PoE and DEC-NN-(gr)(r)BCM (Figure 13, 14); and only minor approximation error for DEC-NN-NPAE [58] (Figure 16). The computation time per agent and the communication rounds are similar for all methods, except DEC-NN-NPAE [58], which remains one to two orders of magnitude more demanding than the proposed methods. Similarly, DEC-NPAE* requires one to two orders of magnitude more computation and communication on both datasets. Overall, inter-agent communication and wall-clock execution time strongly favor DEC-(NN)-(g)PoE and DEC-(NN)-(gr)(r)BCM. These results demonstrate that the proposed nearest neighbor methods achieve significant communication and computation savings without compromising prediction accuracy, making them highly practical for real-time and bandwidth-limited multi-agent systems.

D. Covariance-Based Nearest Neighbor Methods

In Figure 12, 17, we compare the top five methods in terms of prediction accuracy RMSE, NRMSE; and uncertainty quantification NLPD for both datasets D1 and D2. The

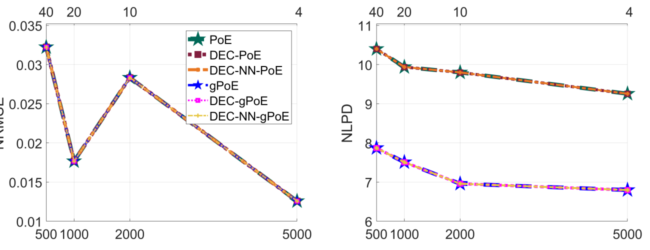

FIGURE 13. **Average NRMSE and NLPD values for four fleet sizes with the PoE-based methods on any graph topology.**

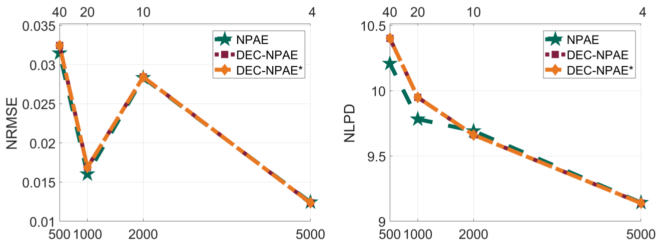

FIGURE 15. **Average NRMSE and NLPD values for four fleet sizes with the NPAE-based methods on any graph topology.**

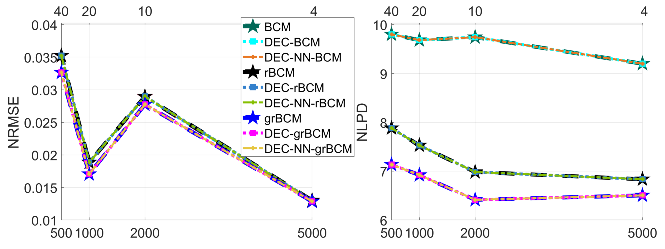

FIGURE 14. **Average NRMSE and NLPD values for four fleet sizes with the BCM-based methods on any graph topology.**

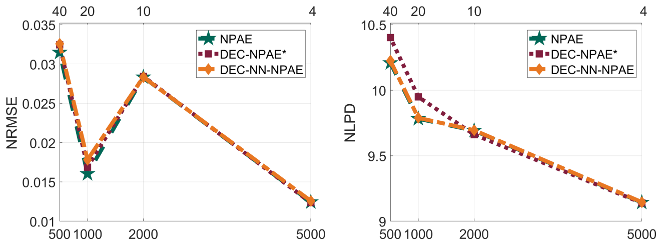

FIGURE 16. **Average NRMSE and NLPD values for four fleet sizes with the NPAE-based methods on a complete graph topology.**

evaluation is conducted under the parsimonious one-hop line graph topology, since, as shown in Section VI-B, network topology does not affect prediction accuracy or uncertainty quantification once convergence is achieved. The comparison includes the DEC-NN-gPoE, DEC-NN-rBCM, DEC-NN-grBCM, DEC-NPAE*, and DEC-NN-NPAE for all four fleet sizes.

For D1 (Figure 12), the results demonstrate high accuracy with low RMSE values for all methods in small fleet sizes (4 and 10 agents). However, as the the number of agents increases, only DEC-NN-rBCM, DEC-NN-grBCM, and DEC-NPAE* recover good accuracy with the latter being the most accurate. Similarly, for uncertainty quantification, all methods provide satisfactorily estimates with low NLPD values for small fleets (4 agents). Yet, as fleet size grows, only DEC-NN-gPoE, DEC-NN-grBCM, and DEC-NPAE* maintain good uncertainty estimates, with DEC-NPAE* performing the best. Notably, DEC-NPAE* is the most accurate method in terms of both RMSE and NLPD, but requires significantly higher execution time and communication overhead to converge (Section VI-C). For D2 (Figure 17), prediction accuracy remains high across all fleet sizes, with consistently low NRMSE values, which is remarkable given the nonstationary nature of the elevation map. In terms of uncertainty quantification, DEC-NN-gPoE, DEC-NN-rBCM, and DEC-NN-grBCM outperform DEC-NPAE* and DEC-NN-NPAE for all fleet sizes. Nevertheless, uncertainty estimates remain high for all methods in D2, indicating that while prediction accuracy is reliable in nonstationary fields, the associated confidence levels are less robust. In summary, the results highlight a trade-off between accuracy and efficiency: while DEC-NPAE* achieves the highest prediction and uncertainty performance, the proposed nearest-neighbor methods achieve a better balance between accuracy, scalability, and communication efficiency, making them more practical for low-resource multi-agent systems.

A qualitative assessment is presented in Table 6 with respect to accuracy, uncertainty quantification, communication overhead, and time scalability. The results indicate that DEC-NN-grBCM emerges as the most balanced decentralized GP prediction method overall. DEC-NPAE* provides high accuracy and reliable uncertainty quantification with reasonable computational demands, but requires significant communication. In contrast, DEC-NN-rBCM achieves good accuracy and scalability with minimal information exchange, but does not provide prediction consistency.

VII. Discussion

In this section, we explicitly address the core assumptions of the proposed methods, including the independence of local datasets (Assumption 3), connectivity requirements (Assumption 1, 7), and federated learning criteria (Assumption 2, 5). We also examine the communication scalability of the proposed methods with respect to network topology and analyze how these assumptions may affect performance in practice. We provide a comparison of the proposed methods in Table 7, and identify trade-offs across application domains.

Table 7 highlights that all methods, with the exception of DEC-NPAE*, require only connected graphs, making them suitable for decentralized systems with limited communication capabilities. The DEC-NPAE* is the only proposed method that requires a complete graph topology, which constrains its applicability in bandwidth-limited networks. With respect to consistency, only DEC-(NN)-grBCM and DEC-NPAE* provide theoretical guarantees, suggesting that they are more robust for applications where high prediction

TABLE 5. Nearest Neighbors, Execution Time, and Communication Rounds of Decentralized Nearest Neighbor Methods for Ground Elevation Dataset D2.

Number of Agents M	Method	Nearest Neighbors	One-hop Line Graph		Two-hop Line Graph		Random Graph	
			Average Time per Agent [s]	Average Comms s^{end}	Average Time per Agent [s]	Average Comms s^{end}	Average Time per Agent [s]	Average Comms s^{end}
4	DEC-NN-PoE	3.2±0.1	0.0803	4.0	0.0429	3.7	0.0426	3.7
	DEC-NN-gPoE		0.0794	4.0	0.0825	3.7	0.0833	3.7
	DEC-NN-BCM		0.0831	4.0	0.0812	3.7	0.0817	3.7
	DEC-NN-rBCM		0.1185	4.0	0.1034	3.7	0.1066	3.7
	DEC-NN-grBCM		0.1214	4.0	0.1161	3.7	0.1215	3.7
	DEC-NN-NPAE [58]		2.8160	77.3	2.5871	71.4	2.5206	71.1
10	DEC-NN-PoE	7.6±0.3	0.0608	8.3	0.0531	7.6	0.0498	7.2
	DEC-NN-gPoE		0.0636	8.3	0.0531	7.6	0.0498	7.2
	DEC-NN-BCM		0.0632	8.3	0.0537	7.6	0.0494	7.2
	DEC-NN-rBCM		0.0872	8.3	0.0810	7.6	0.0778	7.2
	DEC-NN-grBCM		0.0892	8.3	0.0829	7.6	0.0781	7.2
	DEC-NN-NPAE [58]		1.3768	286.7	1.2807	274.0	1.2611	227.9
20	DEC-NN-PoE	14.9±0.5	0.0499	8.6	0.0425	7.8	0.0342	7.3
	DEC-NN-gPoE		0.0512	8.6	0.0433	7.8	0.0379	7.3
	DEC-NN-BCM		0.0580	8.6	0.0492	7.8	0.0390	7.3
	DEC-NN-rBCM		0.0707	8.6	0.0617	7.9	0.0563	7.3
	DEC-NN-grBCM		0.0738	8.6	0.0631	7.9	0.0581	7.3
	DEC-NN-NPAE [58]		0.7127	721.2	0.6117	633.7	0.546.1	576.8
40	DEC-NN-PoE	29.6±1.5	0.0303	8.8	0.0261	8.2	0.0207	7.6
	DEC-NN-gPoE		0.0308	8.8	0.0254	8,2	0.0207	7.6
	DEC-NN-BCM		0.0346	8.8	0.0253	8.2	0.0211	7.6
	DEC-NN-rBCM		0.0419	8.8	0.0370	8.2	0.0266	7.6
	DEC-NN-grBCM		0.0424	8.8	0.0367	8,2	0.0268	7.6
	DEC-NN-NPAE [58]		6.5177	3,708.2	5.7535	3.292.5	5.2069	2,746.8

TABLE 6. Qualitative Assessment of Decentralized GP Methods

Method	RMSE Accuracy	NLPD UQ	Comms	Scalable
DEC-NN-gPoE	Moderate	Moderate	Excellent	Excellent
DEC-NN-rBCM	Excellent	Bad	Excellent	Excellent
DEC-NN-grBCM	Excellent	Excellent	Excellent	Excellent
DEC-NPAE*	Excellent	Excellent	Bad	Moderate
DEC-NN-NPAE [58]	Bad	Moderate	Moderate	Bad

accuracy requirements is essential, at the cost of higher communication demands.

Regarding communication, covariance-based nearest neighbor (CBNN) variants improve scalability by reducing the exchange of information to local neighborhoods. This drastically lowers communication costs compared to the whole network communication requirements of non-NN methods. In terms of federated learning capabilities, most methods are compatible, with the notable exception of DEC-(NN)-grBCM, that cannot preserve federated constraints due to partial dataset exchange for the formation of the communication dataset.

The dataset relationship assumption provides insight into the environments where each method can be applied. Methods such as DEC-(g)PoE and DEC-(r)BCM require strict independence assumptions, limiting their use when agent observations are correlated. By contrast, DEC-(NN)-grBCM explicitly account for conditional independence, making them more flexible in multi-agent sensing scenarios.

We provide some applications that align naturally with our proposed methods for multi-agent systems:

- DEC-(g)PoE and DEC-(r)BCM: Environmental monitoring (e.g., temperature estimation, pollution mapping) where independence assumptions are approximately satisfied and there are no strict consistency requirements.
- DEC-(NN)-grBCM: Distributed multi-agent systems operating in correlated environments (e.g., ocean sensing with overlapping sonar ranges, UAV teams sampling wind fields, power grid monitoring and control), where accounting for conditional independence yields better uncertainty estimates.
- Nearest-neighbor variants (DEC-NN-(g)PoE DEC-NN-(r)BCM): Resource-constrained networks (e.g., swarms

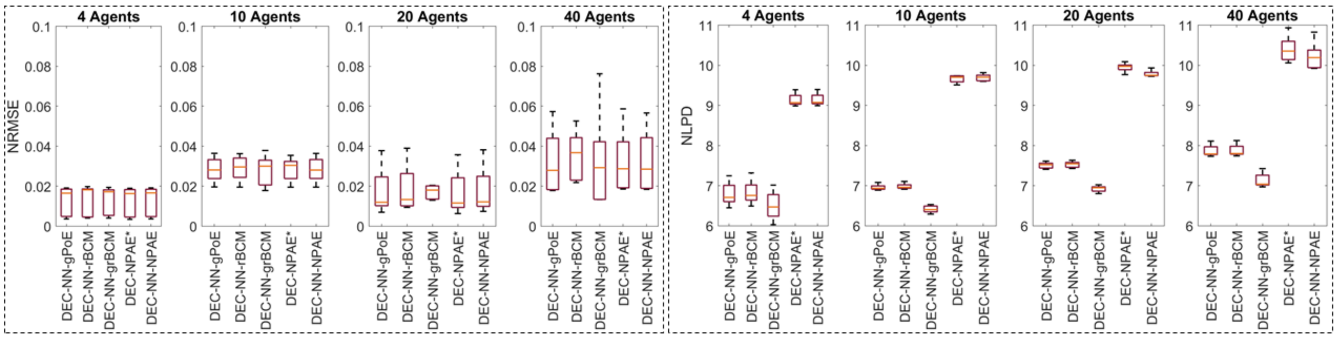

FIGURE 17. Comparison of accuracy and uncertainty quantification for four fleet sizes and 15 replications at $N_t = 100$ unknown locations using $N = 20,000$ observations from the ground elevation dataset in a one-hop line graph topology. Lower NRMSE and NLPD values indicate better accuracy and better uncertainty quantification respectively. The comparison includes the five best decentralized GP prediction methods out of 13. DEC-NN-NPAE was introduced in our previous work [58].

TABLE 7. Comparison of Proposed Decentralized GP Prediction Methods

Method	Connectivity (Assumption 1, 7)	Consistency (Definition 1)	Communication	Federated (Assumption 2, 5)	Dataset Relationship (Assumption 3, 4)
DEC-PoE	Connected	No	Whole	Yes	Independence
DEC-gPoE	Connected	No	Whole	Yes	Independence
DEC-BCM	Connected	No	Whole	Yes	Independence
DEC-rBCM	Connected	No	Whole	Yes	Independence
DEC-grBCM	Connected	Yes	Whole	No	Independence & Cond. Independence
DEC-NN-PoE	Connected	No	NN	Yes	Independence
DEC-NN-gPoE	Connected	No	NN	Yes	Independence
DEC-NN-BCM	Connected	No	NN	Yes	Independence
DEC-NN-rBCM	Connected	No	NN	Yes	Independence
DEC-NN-grBCM	Connected	Yes	NN	No	Independence & Cond. Independence
DEC-NPAE*	Connected & Complete	Yes	Whole	Yes	Independence

of low-cost drones, underwater robots with acoustic communication) where communication and bandwidth must be minimized but approximate aggregation is sufficient.

- DEC-NPAE*: Safety-critical applications (e.g., autonomous driving fleets, air traffic coordination, wildfire detection) where global dataset knowledge and strict consistency are required, while communication infrastructure can support flooding.

The above observations are consistent with our experimental findings in Section VI, where we report communication rounds and execution times, providing evidence of scalability across different graph topologies. The experiments with one-hop line, two-hop line, and Erdős-Rényi random graphs demonstrate how network connectivity directly impacts communication efficiency and execution time. In particular, the proposed CBNN-based methods significantly reduce overhead while maintaining competitive performance, validating their suitability for real-world multi-agent systems.

VIII. Conclusion and Future Work

This paper proposes scalable GP methods for decentralized multi-agent systems. We introduce several techniques that can be used in various applications depending on the fleet size, the computational resources of local agents, and the communication capabilities. Moreover, we introduce a nearest neighbor selection method, namely CBNN, that excludes agents based on statistical correlation from GP prediction. Although CBNN achieves on average 42.5% agent reduction for the sea surface dataset, and 23.9% agent reduction for the ground elevation dataset, it does not sacrifice prediction accuracy, and leads to significant computation and communication reduction. Most of the proposed decentralized GP prediction methods converge to the optimal values without reporting approximation error. The decentralized NPAE-based methods converge with approximation error, yet for DEC-NPAE* the error is insignificant. DEC-NPAE* and DEC-NN-grBCM are the most competitive methods for all fleet sizes both in terms of accuracy and uncertainty quantification, yet DEC-NN-grBCM is also scalable with low communication overhead. Ongoing work is focusing on active learning of GP surrogates for decentralized multi-agent systems.

Appendix

A. Proof of Proposition 2

The prediction mean value of any agent i using PoE yields,

$$\begin{aligned}\mu_{\text{PoE}}(\boldsymbol{x}_*) &= \sigma^2_{\text{PoE}}(\boldsymbol{x}_*)\sum_{i=1}^{M}\beta_i\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*)\\ &= \left(\sum_{i=1}^{M}\beta_i\sigma_i^{-2}(\boldsymbol{x}_*)\right)^{-1}\sum_{i=1}^{M}\beta_i\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*)\\ &= \sum_{i=1}^{M}\sigma_i^{2}(\boldsymbol{x}_*)\sum_{i=1}^{M}\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*). \end{aligned} \tag{23}$$

The prediction mean of the i-th agent using gPoE yields,

$$\begin{aligned}\mu_{\text{gPoE}}(\boldsymbol{x}_*) &= \sigma^2_{\text{gPoE}}(\boldsymbol{x}_*)\sum_{i=1}^{M}\beta_i\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*)\\ &= \left(\sum_{i=1}^{M}\beta_i\sigma_i^{-2}(\boldsymbol{x}_*)\right)^{-1}\sum_{i=1}^{M}\beta_i\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*)\\ &= \left(\sum_{i=1}^{M}\frac{1}{M}\sigma_i^{-2}(\boldsymbol{x}_*)\right)^{-1}\sum_{i=1}^{M}\frac{1}{M}\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*)\\ &= \left(\frac{1}{M}\right)^{-1}\frac{1}{M}\sum_{i=1}^{M}\sigma_i^{2}(\boldsymbol{x}_*)\sum_{i=1}^{M}\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*)\\ &= \sum_{i=1}^{M}\sigma_i^{2}(\boldsymbol{x}_*)\sum_{i=1}^{M}\sigma_i^{-2}(\boldsymbol{x}_*)\mu_i(\boldsymbol{x}_*). \end{aligned} \tag{24}$$

Hence, from (23), (24) $\mu_{\text{PoE}}(\boldsymbol{x}_*) = \mu_{\text{gPoE}}(\boldsymbol{x}_*)$ for all $i \in \mathcal{V}$.

B. Proof of Proposition 8

Observe that the new process is noiseless and for $N \to \infty$ the correlation matches the covariance matrix $\boldsymbol{K}_{\theta,\mu} = \boldsymbol{C}_{\theta,\mu}$. This implies that the interpolation property is satisfied and $\mu_i(\boldsymbol{x}_*) = y_i(\boldsymbol{x}_*)$, where y_i is the exact value with no error $e_{\mu,i} = 0$ for all $i \in \mathcal{V}$. This results in,

$$\begin{aligned}\text{E}[y_i(\boldsymbol{x}_*) \mid y_i(\boldsymbol{X}_i)] &= \text{E}[y_i(\boldsymbol{x}_*) \mid \mu_i(\boldsymbol{X}_i) + e_{\mu,i}(\boldsymbol{X}_i)]\\ &= \text{E}[y_i(\boldsymbol{x}_*) \mid \mu_i(\boldsymbol{X}_i) + 0]\\ &= \mu_i(\boldsymbol{x}_*).\end{aligned}$$

Following the same logic, we can show that the new variance is an exact approximation. Therefore, the random process approximates a GP in the limit.

C. Proof of Lemma 6

First, we show that the separable squared exponential kernel (2) is a monotonically decreasing function. Let any two observations $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and a location of interest $\boldsymbol{x}_*$ with $\|\boldsymbol{x}_* - \boldsymbol{x}_1\|_2^2 > \|\boldsymbol{x}_* - \boldsymbol{x}_2\|_2^2$. The covariance function of the first observations takes the form of,

$$\begin{aligned}k(\boldsymbol{x}_1,\boldsymbol{x}_*) &= \sigma_f^2\exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_{*,d}-x_{1,d})^2}{l_d^2}\right\}\\ &= \sigma_f^2\exp\left\{-\frac{1}{2}\left(\frac{(x_{*,1}-x_{1,1})^2}{l_1^2}+\frac{(x_{*,2}-x_{1,2})^2}{l_2^2}\right)\right\}.\end{aligned} \tag{25}$$

Since the agents collect data in stripes along y-axis and the network topology is a line graph, the length-scales of the covariance function (2) are the same $l_1 = l_2 = l$. Note that this applies only for the CBNN selection subroutine. Thus, the covariance function of the first observation (25) yields,

$$\begin{aligned}k(\boldsymbol{x}_1,\boldsymbol{x}_*) &= \sigma_f^2\exp\left\{-\frac{1}{2}\left(\frac{(x_{*,1}-x_{1,1})^2}{l^2}+\frac{(x_{*,2}-x_{1,2})^2}{l^2}\right)\right\}\\ &= \sigma_f^2\exp\left\{-\frac{1}{2}\frac{\|\boldsymbol{x}_*-\boldsymbol{x}_1\|_2^2}{l^2}\right\}.\end{aligned}$$

Similarly, the covariance function of the second observation,

$$k(\boldsymbol{x}_2,\boldsymbol{x}_*) = \sigma_f^2\exp\left\{-\frac{1}{2}\frac{\|\boldsymbol{x}_*-\boldsymbol{x}_2\|_2^2}{l^2}\right\}.$$

Provided that if $\|\boldsymbol{x}_* - \boldsymbol{x}_1\|_2^2 > \|\boldsymbol{x}_* - \boldsymbol{x}_2\|_2^2$ then $k(\boldsymbol{x}_1,\boldsymbol{x}_*) \leq k(\boldsymbol{x}_2,\boldsymbol{x}_*)$ for all $\boldsymbol{x}_*,\boldsymbol{x}_1,\boldsymbol{x}_2 \in \mathbb{R}^2$, the covariance function $k(\cdot,\cdot)$ used in CBNN calculation (22) is monotonically decreasing with the rate of decrease depending on the signal variance σ_f^2 and the length-scale l.

The CBNN is described by a sub-graph $\mathcal{G}_{\text{NN}} = (\mathcal{V}_{\text{NN}}, \mathcal{E}_{\text{NN}})$, where $\mathcal{V}_{\text{NN}} = \{i \in \mathcal{V} : \|\boldsymbol{x}_i - \boldsymbol{x}_*\|_2 \leq r_{\text{NN}}\}$ for all $\boldsymbol{x}_*$ with r_{NN} the nearest neighbor radius and $\mathcal{E}_{\text{NN}} \subseteq \mathcal{V}_{\text{NN}} \times \mathcal{V}_{\text{NN}}$. In this CBNN selection, the separable squared exponential kernel (2) behaves as a squared exponential kernel with $l_1 = l_2 = l$. This radial decay creates a circle with radius r_{NN} around the location of interest $\boldsymbol{x}_*$ for the selection of nearest neighbors. Note that the radius is a function of the cross-covariance and the user-defined threshold $r_{\text{NN}}([\boldsymbol{k}_{\mu,*}]_i, \eta_{\text{NN}})$, thus inherits a covariance-based dependence. Next, due to fact that the line graph topology includes only connected agents in the circular space, the connectivity in the nearest neighbor graph $\mathcal{G}_{\text{NN}}$ is preserved. Hence, the CBNN selection maintains connectivity.

D. CBNN Worked Example

For a given test location $\boldsymbol{x}_*$, each agent maintains a local cross-covariance $\boldsymbol{k}_{\mu,*} = [k_{\mu,*}^{(1)}\ k_{\mu,*}^{(2)}\ \ldots\ k_{\mu,*}^{(M)}]^\intercal$. The neighbor set is defined as $\mathcal{N}_{\text{CBNN}}(\boldsymbol{x}_*) = \{i \mid k_{\mu,*}^{(i)} \geq \eta_{\text{NN}}\}$, where η_{NN} is the user-defined cross-covariance threshold. Suppose $M = 4$, $\eta_{\text{NN}} = 0.35$, and $\boldsymbol{k}_{\mu,*} = [0.91\ 0.47\ 0.38\ 0.03]^\intercal$. Then, the agents with covariance higher than 0.35 are selected for the CBNN neighborhood $\mathcal{N}_{\text{CBNN}}(\boldsymbol{x}_*) = \{1, 2, 3\}$ and agent 4 is excluded due to negligible contribution.

References

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

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

[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] 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.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. Athena Scientific, 2003.

[6] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[7] X. Wang, S. Mou, and D. Sun, "Improvement of a distributed algorithm for solving linear equations," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 4, pp. 3113–3117, 2016.

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

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

[10] K.-C. Ma, L. Liu, H. K. Heidarsson, and G. S. Sukhatme, "Data-driven learning and planning for environmental sampling," *Journal of Field Robotics*, vol. 35, no. 5, pp. 643–661, 2018.

[11] A. Viseras, T. Wiedemann, C. Manss, *et al.*, "Decentralized multi-agent exploration with online-learning of Gaussian processes," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 4222–4229.

[12] A. Lederer, M. Zhang, S. Tesfazgi, and S. Hirche, "Networked online learning for control of safety-critical resource-constrained systems based on Gaussian processes," in *IEEE Conference on Control Technology and Applications*, 2022, pp. 1285–1292.

[13] 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.

[14] 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.

[15] 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.

[16] G. Pillonetto, L. Schenato, and D. Varagnolo, "Distributed multi-agent Gaussian regression via finite-dimensional approximations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2098–2111, 2018.

[17] 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, 2019, pp. 7850–7857.

[18] 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.

[19] 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.

[20] 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. 103 811, 2021.

[21] 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.

[22] 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.

[23] L. Booth and S. Carpin, "Distributed estimation of scalar fields with implicit coordination," in *International Symposium on Distributed Autonomous Robotic Systems*, Springer, 2022, pp. 466–478.

[24] 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.

[25] K. Masaba and A. Q. Li, "Multi-robot adaptive sampling based on mixture of experts approach to modeling non-stationary spatial fields," in *International Symposium on Multi-Robot and Multi-Agent Systems*, 2023, pp. 191–198.

[26] 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.

[27] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, 2006, pp. 1257–1264.

[28] M. Titsias, “Variational learning of inducing variables in sparse Gaussian processes,” in *Artificial intelligence and statistics*, PMLR, 2009, pp. 567–574.

[29] 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.

[30] V. Tresp, “A Bayesian committee machine,” *Neural computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[31] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

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

[33] Y. Cao and D. J. Fleet, “Generalized product of experts for automatic and principled fusion of Gaussian process predictions,” *arXiv preprint arXiv:1410.7827*, 2014.

[34] F. Bachoc, N. Durrande, D. Rullière, and C. Chevalier, “Some properties of nested Kriging predictors,” *arXiv preprint arXiv:1707.05708*, 2017.

[35] 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.

[36] D. Rullière, N. Durrande, F. Bachoc, and C. Chevalier, “Nested Kriging predictions for datasets with a large number of observations,” *Statistics and Computing*, vol. 28, no. 4, pp. 849–867, 2018.

[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] C. Neves, “Structural health monitoring of bridges: Model-free damage detection method using machine learning,” Ph.D. dissertation, KTH Royal Institute of Technology, 2017.

[39] 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.

[40] 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 Communications*, vol. 37, no. 6, pp. 1291–1306, 2019.

[41] 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.

[42] G. P. Kontoudis and D. J. Stilwell, “Scalable, federated Gaussian process training for decentralized multi-agent systems,” *IEEE Access*, vol. 12, pp. 77 800–77 815, 2024.

[43] J. Cortés, “Distributed Kriged Kalman filter for spatial estimation,” *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2816–2827, 2009.

[44] S. Choi, M. Jadaliha, J. Choi, and S. Oh, “Distributed Gaussian process regression under localization uncertainty,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 3, 2015.

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

[46] 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.

[47] R. B. Gramacy and D. W. Apley, “Local Gaussian process approximation for large computer experiments,” *Journal of Computational and Graphical Statistics*, vol. 24, no. 2, pp. 561–578, 2015.

[48] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, “Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets,” *Journal of the American Statistical Association*, vol. 111, no. 514, pp. 800–812, 2016.

[49] A. O. Finley, A. Datta, B. D. Cook, D. C. Morton, H. E. Andersen, and S. Banerjee, “Efficient algorithms for Bayesian nearest neighbor Gaussian processes,” *Journal of Computational and Graphical Statistics*, vol. 28, no. 2, pp. 401–414, 2019.

[50] L. Wu, G. Pleiss, and J. P. Cunningham, “Variational nearest neighbor Gaussian process,” in *International Conference on Machine Learning*, PMLR, 2022, pp. 24 114–24 130.

[51] I. Grenier, B. Sansó, and J. L. Matthews, “Multivariate nearest-neighbors Gaussian processes with random covariance matrices,” *Environmetrics*, e2839, 2024.

[52] G. Gattiglio, L. Grigoryeva, and M. Tamborrino, “Nearest neighbors GParareal: Improving scalability of Gaussian processes for parallel-in-time solvers,” *arXiv preprint arXiv:2405.12182*, 2024.

[53] Z. Yuan and M. Zhu, “Lightweight distributed Gaussian process regression for online machine learning,” *IEEE Transactions on Automatic Control*, vol. 69, no. 6, pp. 3928–3943, 2024.

[54] 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. 24 730–24 740.

[55] D. R. Burt, C. E. Rasmussen, and M. Van Der Wilk, “Convergence of sparse variational inference in Gaussian processes regression,” *Journal of Machine Learning Research*, vol. 21, no. 131, pp. 1–63, 2020.

[56] C. Park and D. Apley, “Patchwork Kriging for large-scale Gaussian process regression,” *Journal of Machine Learning Research*, vol. 19, no. 7, pp. 1–43, 2018.

[57] X. Yue and R. Kontar, “Federated Gaussian process: Convergence, automatic personalization and multi-fidelity modeling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 6, pp. 4246–4261, 2024.

[58] 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.

[59] J. Liu, S. Mou, and A. S. Morse, “Asynchronous distributed algorithms for solving linear algebraic equations,” *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 372–385, 2018.

[60] F. E. Udwadia, “Some convergence results related to the JOR iterative method for symmetric, positive-definite matrices,” *Applied Mathematics and Computation*, vol. 47, no. 1, pp. 37–45, 1992.

[61] G. H. Golub and C. F. Van Loan, *Matrix computations*, 4th ed. Johns Hopkins University Press, 2013.

[62] G. P. Kontoudis, “Communication-aware, scalable gaussian processes for decentralized exploration,” Ph.D. dissertation, Virginia Tech, 2022.

[63] G. P. Kontoudis and D. J. Stilwell, “Fully decentralized, scalable Gaussian processes for multi-agent federated learning,” *Technical Report, arXiv preprint arXiv: 2203.02865*, 2022.

[64] T. Yang, X. Yi, J. Wu, *et al.*, “A survey of distributed optimization,” *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019.

[65] 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.

[66] V. Yadav and M. V. Salapaka, “Distributed protocol for determining when averaging consensus is reached,” in *Allerton Conference on Communication, Control, and Computing*, 2007, pp. 715–720.

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

[68] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, “Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models,” *International Journal of Robotics Research*, vol. 36, no. 12, pp. 1286–1311, 2017.

[69] J. Guo, Y.-Z. Xie, and A.-C. Qiu, “JOR iterative method for the modeling of MTLs excited by EMP,” *IEEE Antennas and Wireless Propagation Letters*, vol. 15, pp. 536–539, 2015.

[70] NASA JPL, *GHRSST Level 4 MUR Global Foundation Sea Surface Temperature Analysis*, NOAA National Centers for Environmental Information, Online; Accessed via https://doi.org/10.5067/GHGMR-4FJ04 in March, 2021.

[71] T. M. Chin, J. Vazquez-Cuervo, and E. M. Armstrong, “A multi-scale high-resolution analysis of global sea surface temperature,” *Remote sensing of environment*, vol. 200, pp. 154–169, 2017.

[72] T. G. Farr, P. A. Rosen, E. Caro, *et al.*, “The shuttle radar topography mission,” *Reviews of geophysics*, vol. 45, no. 2, 2007.

[73] NASA JPL, *NASA Shuttle Radar Topography Mission Global 30 arc second*, Online; Accessed via https://dwtkns.com/srtm30m/ in September, 2025.

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

[75] W. Chen, R. Khardon, and L. Liu, “Adaptive robotic information gathering via non-stationary Gaussian processes,” *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 405–436, 2024.

George P. Kontoudis (M’22) is an Assistant Professor with the Mechanical Engineering Department, Colorado School of Mines. He received the M.S. and PhD degrees in Mechanical Engineering and Electrical Engineering at Virginia Tech, in 2018 and 2021 respectively. 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.