Decentralized Nested Gaussian Processes for Multi-Robot Systems

George P. Kontoudis and Daniel J. Stilwell

***Abstract*— In this paper, we propose two decentralized approximate algorithms for nested Gaussian processes in multi-robot systems. The distributed implementation is achieved with iterative and consensus methods that facilitate local computations at the expense of inter-robot communications. Moreover, we propose a covariance-based nearest neighbor robot selection strategy that enables a subset of agents to perform predictions. In addition, both algorithms are proved to be consistent. Empirical evaluations with real data illustrate the efficiency of the proposed algorithms.**

I. Introduction

Networks of autonomous vehicles have received considerable attention in recent years, as they can accomplish tasks that cannot be efficiently addressed by a single autonomous vehicle. Major challenges of multi-robot systems include limited computational resources and communication restrictions. In this work, we propose methods for decentralizing Gaussian process (GPs) [1]–[3] so that they can be implemented efficiently on teams of autonomous vehicles. GPs are used in various multi-robot applications [4]–[17]. The major disadvantage of GPs is the poor scalability with the number of observations. Moreover, GPs are not easily decentralized for implementation across multiple autonomous vehicles due to high communication requirements.

Our aim in this work is to synthesize decentralized methodologies that relax the computation and communication requirements of GPs, by performing only local computations and as little information exchange as possible. We propose two approximation techniques of nested pointwise aggregation of GP experts (NPAE) [18]: i) the decentralized NPAE (DEC-NPAE); and ii) the distributed NPAE (DIST-NPAE). In DEC-NPAE, we decentralize the computations by using: i) Jacobi over-relaxation (JOR) to solve a system of linear equations [19]; and ii) discrete-time average consensus (DAC) to compute average values [20]. Next, in DIST-NPAE we leverage advancements in distributed algorithms to solve systems of linear equations (DALE) [21], [22].

Related work: Despite their effectiveness, GPs scale poorly with the number of observations. Particularly, provided N observations, the training entails $\mathcal{O}(N^3)$ computations and the prediction requires $\mathcal{O}(N^2)$ computations. Another limitation for the implementation of GPs in multi-robot systems is the communication. For centralized GPs, every agent has to communicate all observations to a central node. However, excessive communication is challenging in robotic networks, especially in underwater [23] and underground [24] applications. Moreover, robots in distributed networks can pass messages only within a communication range [25] which may vary in space and time [26].

To overcome the computational burden of GPs, multiple techniques have been proposed in the literature. Two major directions are based on global and local approximations [27]. Global approximation methods promote sparsity by using either a subset of M observations or by introducing a set of M pseudo-inputs, where $M \ll N$ [28]–[30]. Sparse GPs have been used in mobile sensor networks to model spatial fields [6]. In [5], a GP with truncated observations in a mobile sensor network is proposed, and in [7] a subset of observations is used for traffic modeling and prediction. These methods require global knowledge of the observations, which increases inter-robot communications. Methods that utilize pseudo-inputs do not retain the interpolation property at the location of observations.

Alternatively, the second research direction to alleviate the GP computation is local approximation methods. These are centralized algorithms with a server-client structure that facilitate the execution of GPs. The main idea is to aggregate local sub-models produced by local subsets of the observations [31]–[34]. In other words, every sub-model makes a local prediction, and then the central node aggregates to a single prediction. In comparison to global approximations, local methods do not require inducing inputs, they distribute the computational load to multiple agents, and they work with all observations. However, it is proved in [35, Prop. 1] that the local methods [31]–[33] are *inconsistent*, i.e. as the observation size grows to infinity, the aggregated predictions do not converge to the true values. Subsequently, the authors in [18] proposed the 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.* [36] introduced a computationally efficient and consistent methodology, termed as generalized robust Bayesian committee machine (GRBCM). However, GRBCM entails additional communication between agents to enrich local datasets with a global random dataset. In addition, both NPAE and GRBCM are centralized techniques, that are not well-suited for multi-robot teams [25].

A decentralized method for the computation of spatio-temporal GPs is proposed in [37]. In [8], a decentralized technique for spatial GPs with localization uncertainty is presented. Both [37] and [8] employ JOR, which requires a strongly complete graph topology, i.e. every agent must communicate to every other agent. That is a conservative

G. P. Kontoudis and D. J. Stilwell are with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA, email: {gpkont,stilwell}@vt.edu.

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

topology and is not common in robotic networks [25]. Essentially, for not strongly complete topologies, JOR entails flooding before every iteration. In flooding each agent broadcasts all input packets to its neighbors [38]. Thus, the communication requirements of JOR are significantly high.

Contributions: The contribution of this paper is fourfold. 1) We synthesize the DEC-NPAE algorithm that decentralizes the computations of NPAE [18]. Essentially, we overcome the main drawback of NPAE—computational complexity in central node—by distributing computations to all robots. 2) To reduce the inter-robot communication required by DEC-NPAE, we propose DIST-NPAE, which integrates a consensus-based, iterative method. 3) A non-arbitrary, covariance-based nearest neighbor (CBNN) selection is proposed to further diminish the information exchange. 4) Both algorithms are proved to produce consistent predictions.

II. Centralized Gaussian Processes

A. Gaussian Processes

Let the observations be modeled by,

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

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

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

where σ_f^2 the signal variance and l_i the length-scale hyperparameter at the i-th direction of the input space. The goal of GPs is to infer the underlying latent function f given the data $\boldsymbol{y} = \{y_n\}_{n=1}^N$, $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$, with N observations.

1) Training: A GP is trained to find the hyperparameters $\boldsymbol{\theta} = \{l_1, \ldots, l_D, \sigma_f^2, \sigma_\epsilon^2\} \in \mathbb{R}^{D+2}$ that maximize the marginal log-likelihood,

$$\mathcal{L} = \ln p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\Big(N \ln(\boldsymbol{y}^\intercal \boldsymbol{C}^{-1} \boldsymbol{y}) + \ln|\boldsymbol{C}| \Big), \tag{3}$$

where $\boldsymbol{C} = \boldsymbol{K} + \sigma_\epsilon^2 I_N$ with $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{N \times N}$ the covariance matrix. Note that (3) includes a constant which vanishes from its gradient.

2) Prediction: After obtaining $\boldsymbol{\theta}$, the predictive distribution of the location of interest $\boldsymbol{x}_* \in \mathbb{R}^D$ conditioned on the data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ yields $p(\boldsymbol{y}_* \mid \mathcal{D}, \boldsymbol{x}_*) \sim \mathcal{N}(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$ with prediction mean and variance,

$$\mu_{\text{full}}(\boldsymbol{x}_*) = \boldsymbol{k}_*^\intercal \boldsymbol{C}^{-1} \boldsymbol{y}, \tag{4}$$

$$\sigma^2_{\text{full}}(\boldsymbol{x}_*) = \sigma_f^2 (k_{**} - \boldsymbol{k}_*^\intercal \boldsymbol{C}^{-1} \boldsymbol{k}_*), \tag{5}$$

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

3) Complexity: The time complexity of the training is $\mathcal{O}(N^3)$ for computing the inverse in (3). Note that only the inverse operation is required in training. This is because the determinant vanishes when computing the derivative of the likelihood (3) for the separable kernel (2). The inverse computation is performed repeatedly in the optimization of (3) to find the hyperparameters $\boldsymbol{\theta}$. Next, we store the inverse $\boldsymbol{C}^{-1}$ and N observations, which results in $\mathcal{O}(N^2 + DN)$ space complexity. For small robots with limited RAM memory capacity, the space complexity may be more restrictive. The prediction mean (4) and variance (5) yield $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ computations respectively for matrix multiplications.

B. Nested Pointwise Aggregation of GP Experts

In this section, we overview the factorized training [33] and NPAE prediction [18]. For M robots, let us partition the dataset $\mathcal{D}$ to M datasets $\{\mathcal{D}_i = \{\boldsymbol{X}_i, \boldsymbol{y}_i\}\}_{i=1}^M$ corresponding to N_i observations for M robots.

Assumption 1. *All local models $\mathcal{M}_i$ are independent.*

1) Training: Provided the independence in Assumption 1 the global marginal likelihood takes the form of,

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) \approx \prod_{i=1}^{M} p_i(\boldsymbol{y}_i \mid \boldsymbol{X}_i, \boldsymbol{\theta}), \tag{6}$$

where $p_i(\boldsymbol{y}_i \mid \boldsymbol{X}_i, \boldsymbol{\theta}) \sim \mathcal{N}(0, \boldsymbol{C}_i)$ is the local marginal likelihood of the i-th robot with $\boldsymbol{C}_i = \boldsymbol{K}_i + \sigma_\epsilon^2 I_{N_i}$ and $\boldsymbol{K}_i = k(\boldsymbol{X}_i, \boldsymbol{X}_i) \in \mathbb{R}^{N_i \times N_i}$. The factorized approximation (6) implies that the covariance matrix is approximated by a block diagonal matrix that results in $\boldsymbol{K}^{-1} \approx \operatorname{diag}(\boldsymbol{K}_1^{-1}, \boldsymbol{K}_2^{-1}, \ldots, \boldsymbol{K}_M^{-1})$. Subsequently, the global marginal log-likelihood is approximated by $\mathcal{L} \approx \sum_{i=1}^M \mathcal{L}_i$ with local marginal log-likelihood,

$$\mathcal{L}_i = \ln p_i(\boldsymbol{y}_i \mid \boldsymbol{X}_i, \boldsymbol{\theta}) = -\frac{1}{2}\Big(N \ln(\boldsymbol{y}_i^\intercal \boldsymbol{C}_i^{-1} \boldsymbol{y}_i) + \ln|\boldsymbol{C}_i| \Big), \tag{7}$$

where $\boldsymbol{C}_i = \boldsymbol{K}_i + \sigma_\epsilon^2 I_{N_i}$. Note that the gradient of the global marginal log-likelihood in factorized training is computed by $\nabla_{\boldsymbol{\theta}} \mathcal{L} = \sum_{i=1}^M \nabla_{\boldsymbol{\theta}} \mathcal{L}_i$ [39], [40].

2) Prediction: Local computations of NPAE for model $\mathcal{M}_i$ include: i) local prediction mean $\mu_i(\boldsymbol{x}_*) \in \mathbb{R}$; ii) i-th entry of the cross-covariance vector $\{k_A(\boldsymbol{x}_*)\}_i \in \mathbb{R}$; and iii) i-th row of the covariance $\operatorname{row}_i\{K_A(\boldsymbol{x}_*)\} \in \mathbb{R}^M$. The local prediction mean yields,

$$\mu_i(\boldsymbol{x}_*) = \mathrm{E}[y(\boldsymbol{x}_*) \mid \boldsymbol{y}(\boldsymbol{X}_i)] = \boldsymbol{k}_{i,*}^\intercal \boldsymbol{C}_i^{-1} \boldsymbol{y}_i \tag{8}$$

where $\boldsymbol{k}_{i,*} = k(\boldsymbol{x}_*, \boldsymbol{X}_i) \in \mathbb{R}^{N_i}$. The cross-covariance and the covariance yield,

$$\{k_A(\boldsymbol{X}_i, \boldsymbol{x}_*)\}_i = \boldsymbol{k}_{i,*}^\intercal \boldsymbol{C}_i^{-1} \boldsymbol{k}_{i,*}, \tag{9}$$

$$\operatorname{row}_i\{\boldsymbol{K}_A(\boldsymbol{X}_i, \boldsymbol{X}_j, \boldsymbol{x}_*)\} = \boldsymbol{k}_{i,*}^\intercal \boldsymbol{C}_i^{-1} \boldsymbol{K}_{ij} \boldsymbol{C}_j^{-1} \boldsymbol{k}_{j,*}, \tag{10}$$

where $\boldsymbol{K}_{ij} = k(\boldsymbol{X}_i, \boldsymbol{X}_j) \in \mathbb{R}^{N_i \times N_j}$, $\boldsymbol{C}_j = \boldsymbol{K}_j + \sigma_\epsilon^2 I_{N_j}$, and $\boldsymbol{k}_{j,*} = (\boldsymbol{X}_j, \boldsymbol{x}_*) \in \mathbb{R}^{N_j}$ for all $j \neq i$, $j = 1, \ldots, M$.

The next step is to aggregate the local models and obtain the aggregated prediction mean and variance,

$$\mu_{\text{agg}}(\boldsymbol{x}_*) = \boldsymbol{k}_A^\intercal \boldsymbol{C}_A^{-1} \boldsymbol{\mu}, \tag{11}$$

$$\sigma^2_{\text{agg}}(\boldsymbol{x}_*) = \sigma_f^2 (k_{**} - \boldsymbol{k}_A^\intercal \boldsymbol{C}_A^{-1} \boldsymbol{k}_A), \tag{12}$$

where $\boldsymbol{C}_A = \boldsymbol{K}_A + \sigma_\epsilon^2 I_N \in \mathbb{R}^{M \times M}$, $\boldsymbol{k}_A = \{k_A(\boldsymbol{X}_i, \boldsymbol{x}_*)\}_{i=1}^M \in \mathbb{R}^M$, and $\boldsymbol{\mu} = \{\mu_i\}_{i=1}^M \in \mathbb{R}^M$.

Definition 1. *Provided N observations, an aggregate GP method with prediction mean $\hat{\mu}$ and full GP prediction mean μ_{full} is consistent if,*

$$\lim_{N\to\infty} \sup E[(\mu_{\text{full}}(\boldsymbol{x}_*) - \hat{\mu}(\boldsymbol{x}_*))^2] \to 0,$$

for all locations of interest $\boldsymbol{x}_*$.

Definition 1 implies that as the number of observations tends to infinity, the mean squared error of the prediction mean of full GP μ_{full} (4) and the aggregated prediction mean of NPAE $\hat{\mu} = \mu_{\text{agg}}$ (11) goes to zero.

Proposition 1. *[35, Proposition 1] For any collection of aggregated prediction mean values $\mu_1(\boldsymbol{x}_*), \ldots, \mu_M(\boldsymbol{x}_*)$ the* NPAE *is consistent.*

3) Complexity: The computation of (7) for training yields $\mathcal{O}(NN_i^2)$ time complexity. The complexity for prediction consists of two parts: i) the local computations (8), (9), (10); and ii) the aggregation at the central node (11), (12). The time complexity for the local computation entails $\mathcal{O}(N_i^2)$ and for the aggregation at the central node $\mathcal{O}(M^3)$. The memory footprint is $\mathcal{O}(N_i^2 + DN_i)$. Since $N_i \ll N$, the time complexity of NPAE $\mathcal{O}(NN_i^2)$ is significantly less than the time complexity of full GP $\mathcal{O}(N^3)$. Similarly, for the space complexity, NPAE needs $\mathcal{O}(N_i^2 + DN_i)$ memory, while GP requires $\mathcal{O}(N^2 + DN)$ space. The time complexity of NPAE in prediction $\mathcal{O}(N_i^2) + \mathcal{O}(M^3)$ is less than the time complexity of full GP $\mathcal{O}(N^2)$. However, the time complexity of NPAE is higher than that of other local aggregation methods [31]–[33], [36], primarily because of the computations at the central node $\mathcal{O}(M^3)$. In the ensuing discussion, we remove the expensive central computation of NPAE by using decentralized iterative techniques.

III. Decentralized Gaussian Processes

In this section, we discuss graph theoretic tools that allow the mathematical representation of networks. Next, we introduce two decentralized techniques to approximate NPAE.

A. Robotic Network

Suppose that a robotic network consists of agents that can only perform local computations. The network is described by an undirected time-varying graph $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$, where $\mathcal{V} = 1, \ldots, M$ is the set of nodes and $\mathcal{E}(t) \subseteq \mathcal{V} \times \mathcal{V}$ the set of edges at time t. Nodes represent robots and edges communication between robots. The neighbors of the i-th robot are denoted $\mathcal{N}_i(t) = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}(t)\}$. The adjacency matrix of $\mathcal{G}(t)$ is denoted $\boldsymbol{A}(t) = [a_{ij}] \in \mathbb{R}^{M\times M}$, where $a_{ij} = 1$ if $(i,j) \in \mathcal{E}(t)$ and $a_{ij} = 0$ otherwise. Similarly, the degree matrix of $\mathcal{G}(t)$ is denoted $\boldsymbol{D}(t) = [d_{ij}] \in \mathbb{R}^{M\times M}$ and is diagonal with $d_i = \sum_{j=1}^{M} a_{ij}$. The graph Laplacian is defined as $\mathcal{L}(t) := \boldsymbol{D}(t) - \boldsymbol{A}(t)$. The maximum degree is denoted $\Delta = \max_i\{\sum_{j\neq i} a_{ij}\}$ and represents the maximum number of neighbors in the graph. The Perron matrix is defined as $\mathcal{P}(t) := I_M - \epsilon\mathcal{L}(t)$, where ϵ is a parameter with range $\epsilon \in (0, 1]$. The maximum shortest distance between any pair of nodes in $\mathcal{G}$ is denoted $\text{diam}(\mathcal{G})$.

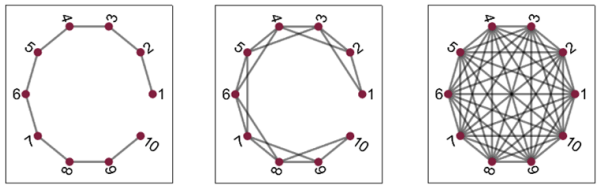


Fig. 1. Graph topologies of multi-robot systems.

If the adjacency matrix A is irreducible, then the graph $\mathcal{G}$ is strongly connected [20]. In addition, a graph $\mathcal{G}$ is strongly complete if every robot can communicate to every other robot in the graph. We consider three network topologies as presented in Fig.1.

Assumption 2. *[22] There exists a positive integer $\gamma \in \mathbb{Z}_{\geq 0}$ such that for all time t the graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}(t) \cup \mathcal{E}(t\gamma + 1) \cup \ldots \cup \mathcal{E}((t+1)\gamma - 1)$ is strongly connected.*

For the local computation of covariance (10) the agents must know the location of observations $\boldsymbol{X}_j$ of all other agents, to find $\boldsymbol{K}_{ij}$, $\boldsymbol{C}_j$, and $\boldsymbol{k}_{j,*}$. This information can be communicated between agents, although explicitly consider the case that it is known *a priori* by all agents.

Assumption 3. *The trajectories $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^M$ and the input locations $\boldsymbol{X} = \{\boldsymbol{X}_i\}_{i=1}^M$ are known to all robots.*

Assumption 4. *All robots know the size of the fleet M.*

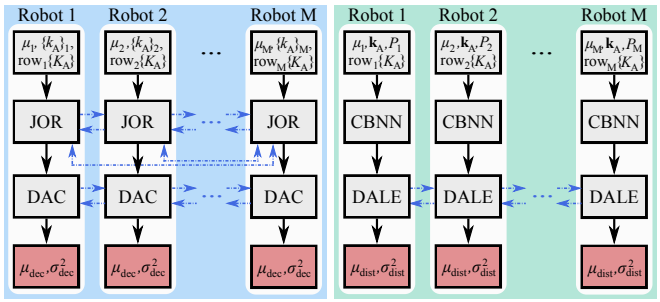
B. Decentralized NPAE

We present DEC-NPAE which combines JOR [19, Ch. 2.4] and DAC [20] to decentralize the computations (11), (12) of NPAE (Fig. 2-(a)). The JOR is an iterative method to solve a system of linear algebraic equations in the form of $\boldsymbol{H}\boldsymbol{z} = \boldsymbol{b}$, 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 $\boldsymbol{z} \in \mathbb{R}^M$ is an unknown vector. More specifically, the i-th robot 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 $z_i \in \mathbb{R}$, the i-th element of the unknown vector $\boldsymbol{z}$. The JOR follows,

$$z_i^{(s+1)} = (1-\omega)z_i^{(s)} + \frac{\omega}{h_{ii}}\left(b_i - \sum_{j\neq i} h_{ij} z_j^{(s)}\right), \quad (13)$$

where $s \in \mathbb{Z}_{\geq 0}$ is the iteration number and $\omega \in (0, 1)$ the relaxation parameter.

Remark 1. *The summation in* (13) *requires communication with all robots, as it is computed over j other than i. This means that each agent must know the update value* (13) *of every other agent $\{\boldsymbol{z}_j^{(s)}\}_{j\neq i}$. That is a major restriction, as it imposes a strongly complete graph (Fig. 1). Although in [8], [37], [41] JOR is used for distributed networks, it is unrealistic for many robotic applications due to limited communication. However, we evaluate the use of JOR, as in some applications with small fleet size, strongly complete networks are feasible. For not strongly complete network*



(a) DEC-NPAE (b) DIST-NPAE

Fig. 2. The structure of DEC-NPAE and DIST-NPAE. Blue dotted lines correspond to communication. (a) DEC-NPAE incorporates Jacobi over-relaxation (JOR) and discrete-time average consensus (DAC). (b) DIST-NPAE makes use of covariance-based nearest neighbors (CBNN) and a distributed algorithm to solve systems of linear equations (DALE).

topologies, distributed flooding is required at every iteration to obtain $\{\boldsymbol{z}_j^{(s)}\}_{j\neq i}$ *and implement* (13)*. The number of inter-robot communications for distributed flooding is the diameter of the graph* $\mathrm{diam}(\mathcal{G})$*. Thus, the total number of iterations yields* $s_{\text{JOR}} = \mathrm{diam}(\mathcal{G}) s_{\text{JOR}}^{(\text{end})}$.

We execute two parallel JOR algorithms with known matrix $\boldsymbol{H} = \boldsymbol{K}_A$ and known vectors: i) $\boldsymbol{b} = \boldsymbol{\mu}$; and ii) $\boldsymbol{b} = \boldsymbol{k}_A$. The first JOR is associated with the prediction mean (11) and the second with the variance (12). Note that $\boldsymbol{K}_A$ is a symmetric and positive definite covariance matrix.

Lemma 1. *[42, Theorem 2] Let the graph* $\mathcal{G}$ *be time-invariant and strongly complete. If* $\boldsymbol{H}$ *is symmetric and positive definite, and* $\omega < 2/M$*, then the JOR converges to the solution for any initialization* $z_i^{(0)}$.

To conserve space we discuss the computation of the prediction mean μ_{agg} (11). The computation of the variance σ^2_{agg} (12) is similar (Alg. 1). We split up the computation of μ_{agg} (11) in two parts. First, each robot computes an element of the unknown vector $z_i = \{\boldsymbol{K}_A^{-1}\boldsymbol{\mu}\}_i \in \mathbb{R}$ (Alg. 1-[lines 18–19]) with the JOR method (13). After every iteration and depending on the network topology, either each agent communicates the computed value $z_i^{(s)} \in \mathbb{R}$ to its neighbors $\mathcal{N}_i$ (Alg. 1-[line 14]), or a distributed flooding is executed to broadcast all computed values $\{\boldsymbol{z}_j^{(s)}\}_{j\neq i} \in \mathbb{R}^{M-1}$ to every other agent (Alg. 1-[line 16]). When JOR (13) converges, each agent computes locally the i-th element of the resulting summation from the multiplication between the vectors $\boldsymbol{k}_A^\intercal$ and $\boldsymbol{K}_A^{-1}\boldsymbol{\mu}$ (11), that is $v_i = \{k_A\}_i z_i$. Second, since all agents have stored a part of the summation v_i, we use the discrete-time average consensus (DAC) that yields,

$$v_i^{(s+1)} = v_i^{(s)} + \epsilon \sum_{j\in\mathcal{N}_i(t)} a_{ij}(t)(v_j^{(s)} - v_i^{(s)}), \tag{14}$$

where ϵ is the parameter of the Perron matrix and $a_{ij}(t)$ is the (i,j)-th entry of the adjacency matrix. After every iteration of DAC (14) (Alg. 1-[lines 24–25]) and irrespective of the network topology, each agent communicates the computed value $v_i^{(s)} \in \mathbb{R}$ to its neighbors $\mathcal{N}_i$ (Alg. 1-[line 23]). Use of

Algorithm 1 DEC-NPAE

Input: $\boldsymbol{X}$, k, M, $\boldsymbol{A}$, $\boldsymbol{x}_*$, η

Output: μ_{dec}, σ^2_{dec}

1: initialize $\omega = 2/M$; $\epsilon = 1/\Delta$
2: **for each** $i \in \mathcal{V}$ **do** ▷ Training
3: $\quad\boldsymbol{y}_i \leftarrow \texttt{measure}(\boldsymbol{X}_i)$
4: $\quad\boldsymbol{\theta} \leftarrow \texttt{factorTraining}(\boldsymbol{y}_i, \boldsymbol{X}_i, k)$ ▷ Section II-B.1
5: **end for**
6: **for each** $i \in \mathcal{V}$ **do** ▷ Prediction
7: $\quad\mu_i \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \boldsymbol{X}_i, \boldsymbol{\theta}, \boldsymbol{y}_i)$ (8)
8: $\quad\{\boldsymbol{k}_A\}_i \leftarrow \texttt{crossCovariance}(\boldsymbol{x}_*, k, \boldsymbol{X}_i, \boldsymbol{\theta})$ (9)
9: $\quad\mathrm{row}_i\{\boldsymbol{K}_A\} \leftarrow \texttt{localCovariance}(\boldsymbol{x}_*, k, \boldsymbol{X}, \boldsymbol{\theta})$ (10)
10: $\quad\boldsymbol{H}_i = \mathrm{row}_i\{\boldsymbol{K}_A\}$; $b_{\mu,i} = \mu_i$; $b_{\sigma^2,i} = \{\boldsymbol{k}_A\}_i$
11: $\quad$initialize $z^{(0)}_{\mu,i} = b_{\mu,i}/\{\boldsymbol{H}\}_{ii}$, $z^{(0)}_{\sigma^2,i} = b_{\sigma^2,i}/\{\boldsymbol{H}\}_{ii}$
12: $\quad$**repeat** ▷ JOR
13: $\quad\quad$**if** graph is strongly complete **then**
14: $\quad\quad\quad$communicate $z^{(s)}_{\mu,i}$, $z^{(s)}_{\sigma^2,i}$ to neighbors $\mathcal{N}_i$
15: $\quad\quad$**else**
16: $\quad\quad\quad\{\boldsymbol{z}^{(s)}_{\mu,j}\}_{j\neq i}$, $\{\boldsymbol{z}^{(s)}_{\sigma^2,j}\}_{j\neq i} \leftarrow \texttt{flooding}$
17: $\quad\quad$**end if**
18: $\quad\quad z^{(s+1)}_{\mu,i} \leftarrow \texttt{JOR}(\omega, \boldsymbol{H}_i, b_{\mu,i}, z^{(s)}_{\mu,i}, \{\boldsymbol{z}^{(s)}_{\mu,j}\}_{j\neq i})$ (13)
19: $\quad\quad z^{(s+1)}_{\sigma^2,i} \leftarrow \texttt{JOR}(\omega, \boldsymbol{H}_i, b_{\sigma^2,i}, z^{(s)}_{\sigma^2,i}, \{\boldsymbol{z}^{(s)}_{\sigma^2,j}\}_{j\neq i})$ (13)
20: $\quad$**until** maximin stopping criterion
21: $\quad$initialize $v^{(0)}_{\mu,i} = \{k_A\}_i z^{(\text{end})}_{\mu,i}$, $v^{(0)}_{\sigma^2,i} = \{k_A\}_i z^{(\text{end})}_{\sigma^2,i}$
22: $\quad$**repeat** ▷ DAC
23: $\quad\quad$communicate $v^{(s)}_{\mu,i}$, $v^{(s)}_{\sigma^2,i}$ to neighbors $\mathcal{N}_i$
24: $\quad\quad v^{(s+1)}_{\mu,i} \leftarrow \texttt{DAC}(\epsilon, v^{(s)}_{\mu,i}, \{\boldsymbol{v}^{(s)}_{\mu,j}\}_{j\in\mathcal{N}_i}, \mathcal{N}_i)$ (14)
25: $\quad\quad v^{(s+1)}_{\sigma^2,i} \leftarrow \texttt{DAC}(\epsilon, v^{(s)}_{\sigma^2,i}, \{\boldsymbol{v}^{(s)}_{\sigma^2,j}\}_{j\in\mathcal{N}_i}, \mathcal{N}_i)$ (14)
26: $\quad$**until** maximin stopping criterion
27: $\quad\mu_{\text{dec}} = M v^{(\text{end})}_{\mu,i}$ ▷ Local Mean
28: $\quad\sigma^2_{\text{dec}} = \sigma^2_f(k(\boldsymbol{x}_*, \boldsymbol{x}_*) - M v^{(\text{end})}_{\sigma^2,i})$ ▷ Local Variance
29: **end for**

consensus protocols implicitly requires that each robot can distributively determine convergence in the robotic network. In other words, just because an agent converged does not imply that the network has reached consensus. We employ a maximin stopping criterion [43] to locally detect convergence in the network for JOR (Alg. 1-[line 20]) and DAC (Alg. 1-[line 26]). When DAC (14) converges, each robot knows $v_i^{(\text{end})} = (1/M)\boldsymbol{k}_A^\intercal \boldsymbol{K}_A^{-1}\boldsymbol{\mu}$. Finally, using Assumption 4 the robots compute the mean (11) with $\mu_{\text{dec}} = M v_i^{(\text{end})}$.

Lemma 2. *[20, Theorem 2], [44, Cor. 5.2] Let Assumption 2 hold. If* $0 < \epsilon < 1/\Delta$*, then the distributed consensus algorithm* (14) *converges to the average for any initialization* $v_i^{(0)}$ *with convergence speed* $T_n(\epsilon) = \mathcal{O}(M^3 \log(M/\epsilon))$.

Proposition 2. *Let the graph* $\mathcal{G}$ *be either time-invariant strongly compete or time-invariant strongly connected during the JOR iterations and let Assumption 2 hold during the DAC iterations. In addition, let Assumption 3 and Assumption 4 hold throughout the approximation. If* $\omega < 2/M$, $0 < \epsilon < 1/\Delta$*, then the* DEC-NPAE *is consistent for any initialization.*

Proof: The proof is a direct consequence of Proposition 1, Lemma 1, and Lemma 2.

C. Distributed NPAE

We introduce DIST-NPAE that uses DALE to distribute the computations (11), (12) of NPAE (Fig. 2-(b)). The DALE is an iterative method to solve a system of linear equations

Algorithm 2 DIST-NPAE

Input: $\boldsymbol{X}$, k, M, $\boldsymbol{A}$, $\boldsymbol{x}_*$, η, η_{NN}

Output: μ_{dist}, σ^2_{dist}

1: **for each** $i \in \mathcal{V}$ **do** ▷ Training
2: $\quad \boldsymbol{y}_i \leftarrow \texttt{measure}(\boldsymbol{X}_i)$
3: $\quad \boldsymbol{\theta} \leftarrow \texttt{factorTraining}(\boldsymbol{y}_i, \boldsymbol{X}_i, k)$ ▷ Section II-B.1
4: **end for**
5: **for each** $i \in \mathcal{V}$ **do** ▷ Prediction
6: $\quad \mu_i \leftarrow \texttt{localMean}(\boldsymbol{x}_*, k, \boldsymbol{X}_i, \boldsymbol{\theta}, \boldsymbol{y}_i)$ (8)
7: $\quad \boldsymbol{k}_A \leftarrow \texttt{crossCovariance}(\boldsymbol{x}_*, k, \boldsymbol{X}, \boldsymbol{\theta})$ (9)
8: $\quad \text{row}_i\{\boldsymbol{K}_A\} \leftarrow \texttt{localCovariance}(\boldsymbol{x}_*, k, \boldsymbol{X}, \boldsymbol{\theta})$ (10)
9: $\quad$ **for each** $i \in \mathcal{V}$ **do** ▷ Nearest Neighbor
10: $\quad\quad$ **if** $\{k_A\}_i < \eta_{\text{NN}}$ **then**
11: $\quad\quad\quad$ delete $\{\boldsymbol{k}_A\}_i$ from $\boldsymbol{k}_A$
12: $\quad\quad\quad$ delete $\{\text{row}_i\{\boldsymbol{K}_A\}\}_i$ from $\text{row}_i\{\boldsymbol{K}_A\}$
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad \boldsymbol{H}_i = \text{row}_i\{\boldsymbol{K}_A\}$; $b_{\mu,i} = \mu_i$; $\boldsymbol{b}_{\sigma^2} = \boldsymbol{k}_A$
16: $\quad \boldsymbol{P}_i \leftarrow \texttt{kernelProjection}(M, \boldsymbol{H}_i)$
17: $\quad$ initialize $\boldsymbol{z}^{(0)}_{\mu,i} = b_{\mu,i} \oslash \boldsymbol{H}_i$, $\boldsymbol{z}^{(0)}_{\sigma^2,i} = \boldsymbol{b}_{\sigma^2} \oslash \boldsymbol{H}_i$
18: $\quad$ **repeat** ▷ DALE
19: $\quad\quad$ communicate $\boldsymbol{z}^{(s)}_{\mu,i}$, $\boldsymbol{z}^{(s)}_{\sigma^2,i}$ to neighbors $\mathcal{N}_i$
20: $\quad\quad \boldsymbol{z}^{(s+1)}_{\mu,i} \leftarrow \texttt{DALE}(\boldsymbol{P}_i, \boldsymbol{H}_i, b_{\mu,i}, \{\boldsymbol{z}^{(s)}_{\mu,j}\}_{j\in\mathcal{N}_i}, \mathcal{N}_i)$ (15)
21: $\quad\quad \boldsymbol{z}^{(s+1)}_{\sigma^2,i} \leftarrow \texttt{DALE}(\boldsymbol{P}_i, \boldsymbol{H}_i, b_{\sigma^2,i}, \{\boldsymbol{z}^{(s)}_{\sigma^2,j}\}_{j\in\mathcal{N}_i}, \mathcal{N}_i)$ (15)
22: $\quad$ **until** maximin stopping criterion
23: $\quad \mu_{\text{dist}} = \boldsymbol{k}_A^\intercal \boldsymbol{z}^{(\text{end})}_{\mu,i}$ ▷ Local Mean
24: $\quad \sigma^2_{\text{dist}} = \sigma^2_f(k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_A^\intercal \boldsymbol{z}^{(\text{end})}_{\sigma^2,i})$ ▷ Local Variance
25: **end for**

with identical setup to JOR $\boldsymbol{H}\boldsymbol{z} = \boldsymbol{b}$, where $\boldsymbol{H}$ is a known matrix, $\boldsymbol{b}$ a known vector, and $\boldsymbol{z}$ an unknown vector. The i-th robot knows: i) i-th row of $\boldsymbol{H}_i = \text{row}_i\{\boldsymbol{H}\} \in \mathbb{R}^{1\times M}$; and ii) i-th element of the $b_i \in \mathbb{R}$. In addition, DALE is formulated as a consensus problem, where the goal for all robots is to obtain the same solution $\boldsymbol{z}_i \in \mathbb{R}^M$ and not just an element of the unknown vector as in JOR. The DALE follows,

$$\boldsymbol{z}_i^{(s+1)} = \boldsymbol{H}_i^\intercal(\boldsymbol{H}_i\boldsymbol{H}_i^\intercal)^{-1}b_i + \frac{1}{\text{card}(\mathcal{N}_i(t))}\boldsymbol{P}_i \sum_{j\in\mathcal{N}_i(t)} \boldsymbol{z}_j^{(s)}, \tag{15}$$

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 $\boldsymbol{H}_i$.

Remark 2. *Notably, the i-th robot using DALE* (15) *exchanges information only with its neighbors $\mathcal{N}_i$ and not with the whole network (see in contrast Remark 1 for JOR). In addition, DALE is concurrently a consensus algorithm and updates the whole vector $\boldsymbol{z}_i^{(s)} \in \mathbb{R}^M$, while JOR updates just the corresponding entry $z_i^{(s)} \in \mathbb{R}$. This concurrent operation of DALE makes it equivalent to the operation of both JOR and DAC. To this end, the communication events of* DIST-NPAE *are expected to be significantly reduced, compared to* DEC-NPAE*. A qualitative comparison of communication events for* DEC-NPAE *and* DIST-NPAE *is shown in Fig. 2.*

To ensure that $\boldsymbol{H}$ is full row rank, we introduce the covariance-based nearest neighbor (CBNN). Note that Assumption 3 allows the local computation of the cross-covariance vector $\boldsymbol{k}_A$ (9) and not just the i-th entry $\{\boldsymbol{k}_A\}_i$. In addition, DALE can be executed in a time-varying network under Assumption 2. The main idea is to use these two observations in order to select which robots should be involved in the aggregation. In other words, instead of specifying an arbitrary radius to determine the nearest neighbors, we identify the non-zero elements of the cross-covariance vector $\boldsymbol{k}_A(\boldsymbol{x}_*)$ and use only these robots $M_{\text{NN}} \in [2, M]$ to the aggregation. The CBNN selection method is executed before the DALE (15) and modifies the cross-covariance $\boldsymbol{k}_A$ and covariance $\boldsymbol{K}_A$ accordingly (Alg. 2-[lines 9–14]).

TABLE I

TIME AND SPACE COMPUTATIONAL COMPLEXITY

Complexity	FULL-GP	NPAE local	NPAE global	DEC- & DIST-NPAE local
Time (Training)	$\mathcal{O}(N^3)$	$\mathcal{O}(NN_i^2)$		$\mathcal{O}(NN_i^2)$
Time (Prediction)	$\mathcal{O}(N^2)$	$\mathcal{O}(N_i^2)$	$\mathcal{O}(M^3)$	$\mathcal{O}(N_i^2)$
Space	$\mathcal{O}(N^2 + DN)$	$\mathcal{O}(N_i^2 + DN_i)$		$\mathcal{O}(N_i^2 + DN_i)$

Similarly to DEC-NPAE, we execute two parallel DALE algorithms with known matrix $\boldsymbol{H} = \boldsymbol{K}_A$ and known vectors: i) $\boldsymbol{b} = \boldsymbol{\mu}$; and ii) $\boldsymbol{b} = \boldsymbol{k}_A$. The first DALE is associated with the prediction mean μ_{agg} (11) and the second with the variance σ^2_{agg} (12). To conserve space we discuss only the computation of the mean (11), yet the variance (12) is similar (Alg. 2). Since DALE is concurrently an iterative and a consensus method, the DIST-NPAE is an one step process. To this end, each robot computes the unknown vector $\boldsymbol{z}_i = \{\boldsymbol{K}_A^{-1}\boldsymbol{\mu}\}_i \in \mathbb{R}^M$ (Alg. 2-[lines 20–21]) using DALE (15). Next, at every iteration the computed vector $\boldsymbol{z}_i^{(s)} \in \mathbb{R}^M$ is communicated to the neighbor set $\mathcal{N}_i$ (Alg. 2-[line 19]). When (15) converges, every robot has access to the same vector. Next, we exploit Assumption 3 to compute the directly aggregation as $\mu_{\text{dist}} = \boldsymbol{k}_A^\intercal \boldsymbol{z}_i^{(\text{end})}$.

Proposition 3. *Let Assumption 2 and 3 hold. Then, the* DIST-NPAE *is consistent for any initialization of DALE.*

Proof: (Sketch) It can be shown that the separable squared exponential kernel (2) *is a monotonically decreasing function that characterizes the CBNN with an ellipse. Observe that the selected graph topologies include only connected robots in the elliptical space, preserving the connectivity in the nearest neighbor graph $\mathcal{G}_{NN}$. Hence, the* DIST-NPAE *with CBNN maintains strong connectivity. The rest is a direct consequence of DALE convergence [22, Theorem 3].*

D. Complexity and Convergence

A comparison of time and space complexity is presented in Table I. Since we employ iterative methods, our interest is focused on communication and accuracy. Essentially, the number of iteration upon convergence is identical to inter-robot communication events. In other words, the cumulative number of iterations (for DEC-NPAE $s_{\text{dec}} = s_{\text{JOR}} + s_{\text{DAC}}$ and for DIST-NPAE $s_{\text{dist}} = s_{\text{DALE}}$) is the required communications per agent. In addition, the CBNN in DIST-NPAE diminishes the communications per agent when the nearest neighbors M_{NN} are less then the size of the fleet M.

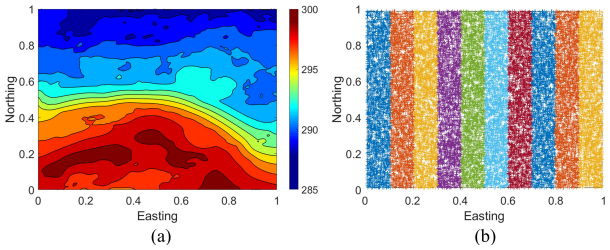


Fig. 3. (a) SST field [45]; (b) Observations of each robot for $M = 10$.

IV. Numerical Experiments

Numerical experiments are performed to illustrate the efficacy of our methods. We use a real-world dataset of sea surface temperature (SST) [45], [46]. We extract $122,500$ SST values from $(36.4^{\text{o}}, -73.0^{\text{o}})$ to $(40.0^{\text{o}}, -69.4^{\text{o}})$ measured in Kelvins. The area corresponds to $400\,\text{km} \times 400\,\text{km}$ of the Atlantic ocean and for demonstration is normalized over $[0, 1]^2$ (Fig. 3-(a)). Additionally, we add iid noise $\epsilon \sim \mathcal{N}(0, 0.25)$ to the observations (1). We use $20,000$ observations, equally distributed for seven fleet sizes $\boldsymbol{M} = \{2, 4, 5, 10, 20, 25, 40\}$ (Fig. 3-(b)). We employ six techniques over 100 prediction points: i) FULL-GP; ii) NPAE; iii) DEC-NPAE with complete graph; iv) DEC-NPAE with path graph; v) DIST-NPAE with path graph; and vi) DIST-NPAE with augmented path graph, where each graph type is shown in Fig. 1. For every scenario we perform 15 replications to remove the effect of random assignment of data.

The quality assessment is accomplished with four metrics. The root mean square error $\text{RMSE} = [1/N \sum_{i=1}^{N} (\mu(\boldsymbol{x}_*) - y(\boldsymbol{x}_*))^2]^{1/2}$ assesses the accuracy. The negative log predictive density $\text{NLPD} = -1/N \sum_{i=1}^{N} \log p(\hat{\boldsymbol{y}}_* \mid \mathcal{D}, \boldsymbol{x}_*)$ characterizes the error of the prediction mean and variance, where $p(\hat{\boldsymbol{y}}_* \mid \mathcal{D}, \boldsymbol{x}_*)$ is the predictive distribution [47]. The mean variance error $\text{MVE} = 1/N \sum_{i=1}^{N} (\sigma^2(\boldsymbol{x}_*) - \sigma^2_{\text{agg}}(\boldsymbol{x}_*))$ evaluates the prediction variance with respect to the variance of NPAE. Lastly, we count the communications per agent as the number of iterations s_{dec} and s_{dist}. Demonstration code can be found at: github.com/gkontoudis/decentralized-GP.

In Fig. 4-(a), we show the average RMSE values for all 15 replications. Since our algorithms approximate NPAE, the lowest possible RMSE values are that of NPAE (dotted red). We observe that for fleet size equal or greater to 25 robots, DIST-NPAE with path graph (dash-dotted blue) and with augmented path graph (dashed pink) are inaccurate, as they produce high RMSE values. When the fleet comprises of 20 agents or less, then DIST-NPAE offers competitive predictions regardless of the neighborhood size. DEC-NPAE with path graph (dashed green) and DEC-NPAE with strongly complete graph (dotted maroon) produce identical and accurate predictions for all number of agents.

In Fig. 4-(c), we depict the average MVE. Both DIST-NPAE approximate the variance with reasonable MVE values. The DEC-NPAE approximates very accurately the variance of NPAE for all number of agents.

Next, we present the average communications per agent for all 15 trials in Fig. 4-(d). Note that communications per

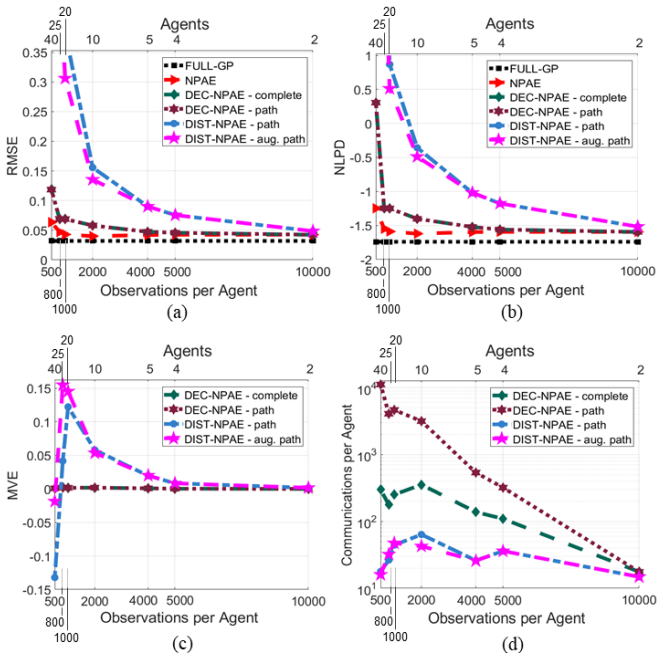


Fig. 4. Approximate methods over $20,000$ observations of seven fleet sizes.

agent is logarithmically scaled and thus any difference in the plot is major. Although DEC-NPAE outperforms DIST-NPAE both in accuracy and uncertainty quantification (see Fig. 4-(a)–(c)), we observe that DEC-NPAE requires substantially more communications. More specifically, the comparison of same path graph topologies (maroon dotted and blue dash-doted) reveals that DEC-NPAE entails at least one order of magnitude more information exchange to converge. Notably, even conservative network topologies with strongly complete graphs of DEC-NPAE (green dashed) entail at least double communications when compared to a much weaker topology, the path graph with DIST-NPAE (pink dashed). Alleviation in communication of DIST-NPAE is occurred due to the CBNN selection strategy and the DALE algorithm. As a result, a trade-off exists between prediction accuracy and communications. For challenging communication environments, DIST-NPAE offers a distributed solution that advocates acceptable predictions with less information exchange than competing methods. In contrary, when communication is unrestricted, DEC-NPAE produces accurate predictions.

V. Conclusion

The two proposed decentralized algorithms, DEC-NPAE and DIST-NPAE, cover a broad spectrum of decentralized GPs in robotic applications. Both methods are computationally efficient using only local computation of NPAE and proved to be consistent under specific graph topologies. The DEC-NPAE produces almost identical predictions to NPAE for all graph topologies regardless of fleet size, yet entails excessive inter-robot communication. On the other hand, DIST-NPAE with CBNN converges with significantly less information exchange and produces satisfactorily predictions for small and medium fleet size.

REFERENCES

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

[2] 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/.

[3] N. Cressie, *Statistics for spatial data*, 2nd ed. Wiley, 1993.

[4] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] R. Allamraju and G. Chowdhary, "Communication efficient decentralized Gaussian process fusion for multi-UAS path planning," in *American Control Conference*, 2017, pp. 4442–4447.

[10] 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.

[11] W. Zhi, L. Ott, R. Senanayake, and F. Ramos, "Continuous occupancy map fusion with fast Bayesian Hilbert maps," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 4111–4117.

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

[13] Z. Yuan and M. Zhu, "Communication-aware distributed Gaussian process regression algorithms for real-time machine learning," in *American Control Conference*, 2020, pp. 2197–2202.

[14] 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.

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

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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] L. Lanbo, Z. Shengli, and C. Jun-Hong, "Prospects and problems of wireless communication for underwater sensor networks," *Wireless Communications and Mobile Computing*, vol. 8, no. 8, pp. 977–994, 2008.

[24] K. Alexis, "Resilient autonomous exploration and mapping of underground mines using aerial robots," in *IEEE International Conference on Advanced Robotics*, 2019, pp. 1–8.

[25] 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.

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

[27] 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*, 2020.

[28] 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.

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

[30] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in *Conference on Uncertainty in Artificial Intelligence*, 2013, p. 282.

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

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

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

[34] 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.

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

[36] 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.

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

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] JPL MUR MEaSUREs Project, "GHRSST Level 4 MUR Global Foundation Sea Surface Temperature Analysis," Ver. 4.1. PO.DAAC, CA, USA; https://doi.org/10.5067/GHGMR-4FJ04, 2015, online; accessed March, 2021.

[46] 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.

[47] J. Quinonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, and B. Schölkopf, "Evaluating predictive uncertainty challenge," in *Machine Learning Challenges Workshop*, 2005, pp. 1–27.