applied sciences

MDPI

Article

Global-Entropy Driven Exploration with Distributed Models under Sparsity Constraints

Christoph Manss *,† and Dmitriy Shutin †

German Aerospace Center, Institute of Communication and Navigation, Münchener Straße 20, 82234 Wessling, Germany; dmitriy.shutin@dlr.de

* Correspondence: Christoph.Manss@dlr.de

† These authors contributed equally to this work.

Received: 24 August 2018; Accepted: 19 September 2018; Published: 22 September 2018

Abstract: This paper focuses on exploration when using different data distribution schemes and ADMM as a solver for swarms. By exploration, we mean the estimation of new measurement locations that are beneficial for the model estimation. In particular, the different distribution schemes are splitting-over-features or heterogeneous learning and splitting-over-examples or homogeneous learning. Each agent contributes a solution to solve the joint optimization problem by using ADMM and the consensus algorithm. This paper shows that some information is unknown to the individual agent, and thus, the estimation of new measurement positions is not possible without further communication. Therefore, this paper shows results for how to distribute only necessary information for a global exploration. We show the benefits between the proposed global exploration scheme and benchmark exploration schemes such as random walk and systematic traversing, i.e., meandering. The proposed waypoint estimation methods are then tested against each other and with other movement methods. This paper shows that a movement method, which considers the current information within the model, is superior to the benchmark movement methods.

Keywords: multi-agent system; swarm exploration; information gathering; optimal experiment design; distributed systems

1. Introduction

Multi-agent systems or swarms have proven to be advantageous in hazardous and inhospitable environments, such as emergency scenarios, natural disasters and extraterrestrial scenarios [1]. Humans operate these robots from a distance and are not put into danger while getting valuable information in a reconnaissance mission or the search for extraterrestrial life. Swarms precipitate in the information gathering process by their spatial distribution and by their larger sensor aperture [2,3]. Especially for coverage problems, swarms are very useful [4–6] and have proven to be advantageous. It is obvious to see that by cooperation, swarms can speed up their mission task significantly. The gain in speed can either be due to performing parallel measurements or due to splitting the computational load among the agents [7]. Additionally, cooperation can open a way for more autonomy in such tasks and as a consequence reduce the mission costs [3].

To process the distributed data cooperatively, two different distribution strategies can be distinguished. The first strategy follows a homogeneous distribution paradigm [8,9], where the swarm members have their own measurements and agree on a commonly-known model, which is then cooperatively estimated. This paradigm is also referred to as SOE, because every agent has an individual example of a common estimate.The second strategy uses a heterogeneous distribution paradigm [8,9], also called SOF. In SOF, every agent has the knowledge of all measurements, but only processes parts of the global model. Thus, it can be seen as a splitting into different feature sets.

Both approaches have been studied for estimation tasks [8–10]. However, a swarm can also cooperate by using the obtained information about their task for a cooperative navigation. Nowadays, cooperative information-driven navigation with swarms is the next key step for more autonomy in swarm systems, especially for SLAM algorithms, but also for information-gathering and exploration tasks. In [11], the authors set up a swarm of three mobile robots. One robot was able to manipulate objects in the environment with a robotic arm. The other two agents explored the a priori unknown environment and detected the positions of desired objects for the robot equipped with the manipulator. The two exploring robots explored the environment faster than just a single robot. A decision-theoretic approach, which takes uncertainty information of the agent's sensors into account, for exploring maps was presented in [12]. The authors estimated the map with a Rao-Blackwellized particle filter and performed exploration by maximizing the information gain, when a specific action was performed, i.e., the robot's movement. To estimate the entropy, the authors used a weighted sum of the posterior of their estimated particles. This approach was also tested in experiments, and the authors showed that this approach was more effective compared to non-information-driven approaches. A similar approach was described in [13], where the authors used a combination of the joint entropy of the map and the gained entropy along a path as a metric for exploration. All these methods focused on entropy-driven approaches for estimating a map. The authors did not consider other processes behind this, such as magnetic fields, gas concentrations or temperatures. Thus, we focus in this paper more on the exploration of such processes instead of the exploration of maps, and consequently, we assume in the following that an obstacle map is known or has been estimated before. Furthermore, we focus on distributed systems, which exchange their information to explore cooperatively.

In this paper, we aim to estimate the spatial distribution of a static process such as a height profile or a magnetic field intensity in an area [14]. Static means that this process possesses no variation in time. The exploration of dynamic processes is therefore not considered in this work. Static processes are of particular interest for SAR applications [15] or height profile estimation in general [16]. Therefore, we aim to represent these processes with a weighted sum of static point spread functions, i.e., kernels, which are also referred to as kernel methods. The contribution of this work is manifold and is summarized in the following.

- We propose a swarm exploration system, which processes the obtained sensor measurements online and processes these distributively.
- We apply two distribution paradigms: SOE and SOF.
- We present how to achieve a joint entropy of a distributed estimation by means of a consensus algorithm.
- We show the performance gain of our system in simulations with real data.

The structure of this paper is as follows. In Section 2, we first show the centralized approach followed by the two distribution paradigms: SOE and SOF. Afterward, we present the main contribution of this paper in Section 3, where we show the entropy-driven exploration schemes for a centralized, SOE and SOF approach. In Section 4, we present the performed simulations. Finally, we draw the conclusions in Section 5.

Throughout the paper, we use the following notation. Matrices are written in capital bold letters $\boldsymbol{\Gamma}, \boldsymbol{P}, \ldots$. Vectors are written in lower case and bold $\boldsymbol{\gamma}, \boldsymbol{p}, \ldots$. Constant scalar values are capital letters, e.g., Γ, P, and scalars are lowercase $\gamma, p, \ldots$. We also use sets, which are denoted in calligraphic letters $\mathcal{P}, \ldots$.

2. Signal Models for Static Process Estimation

We begin by formulating the inference model in a centralized setting. Then, two approaches to a distributed implementation of the inference procedure are considered.

2.1. Centralized Signal Model

Let us consider an agent equipped with a sensor for measuring a spatial process in the environment, which we consider to be two-dimensional. Our goal is to reconstruct a spatial function $f : \mathbb{R}^2 \mapsto \mathbb{R}$ that models this process using collected measurement samples. Formally, we assume that the agent collects M samples $y[m] \in \mathcal{Y} \subset \mathbb{R}$, $m = 1, \ldots, M$, of the process at the positions $\boldsymbol{x}[m] \in \mathcal{X} \subset \mathbb{R}^2$, $m = 1, \ldots, M$. The notation $[m]$ denotes a measurement index taken by the agent. The positions are estimated using some localization system. We assume that the localization problem has been solved using, e.g., SLAM algorithms [17], GNSS or any other localization system. Thus, a measurement is represented with a pair $\{\boldsymbol{x}[m], y[m]\}$.

We will assume the function f to be sufficiently smooth and model it as a weighted superposition of known and fixed kernels $\varphi(\boldsymbol{x}, \boldsymbol{x}') : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. To be able to formulate the problem in a convenient matrix-vector form, we collect all measurements $\{y[m], \boldsymbol{x}[m]\}$, $m = 1, \ldots, M$, as follows: $\boldsymbol{y} \widehat{=} [y[1], \ldots, y[M]]^T$ and $\boldsymbol{X}_M \widehat{=} [\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]]^T$. Now, we define a kernel vector: $\boldsymbol{\varphi}(\boldsymbol{X}_M, \boldsymbol{x}) \widehat{=} [\varphi(\boldsymbol{x}[1], \boldsymbol{x}), \ldots, \varphi(\boldsymbol{x}[M], \boldsymbol{x})]^T$, which collects the values of the kernel function $\varphi(\boldsymbol{x}, \boldsymbol{x}')$ evaluated at all measurement locations $\boldsymbol{X}_M$. The kernel vectors $\boldsymbol{\varphi}(\boldsymbol{X}_M, \boldsymbol{x})$ can then be evaluated at different locations $\boldsymbol{x} \in \mathcal{X}$.

Considering N arbitrary locations $\boldsymbol{X}_N = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T$, we define a design matrix $\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N) = [\boldsymbol{\varphi}(\boldsymbol{X}_M, \boldsymbol{x}_1), \ldots, \boldsymbol{\varphi}(\boldsymbol{X}_M, \boldsymbol{x}_N)] \in \mathbb{R}^{M \times N}$.

The spatial measurements made by the agents are then approximated as a linear combination of the columns of the design matrix $\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)$ using the following model:

$$\boldsymbol{y} = \boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)\boldsymbol{w} + \boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^N$ is a weight vector and $\boldsymbol{\epsilon} \in \mathbb{R}^M$ is additive noise. In the following, we will assume that elements of $\boldsymbol{\epsilon}$ are iid samples from a zero-mean normal distribution with precision λ, i.e., $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \lambda^{-1}\boldsymbol{I})$.

In the considered inference framework, we assume the weights $\boldsymbol{w}$ to be sparse. In other words, we assume that many elements in $\boldsymbol{w}$ are zero. The inclusion of this constraint serves the following purposes. First of all, it acts as a problem regularization, especially when the number of measurements is small, i.e., $M \ll N$. Second, the sparsity constraints naturally permits us to keep only relevant columns in the design matrix $\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)$ even when $M > N$, i.e., the resulting model effectively "compresses" the measured data and represents it in a compact form.

The sparse estimation of the weights $\boldsymbol{w}$ under the measurement model (1) can be solved optimally by means of a LASSO optimization problem [18]:

$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{\lambda}{2} \|\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)\boldsymbol{w} - \boldsymbol{y}\|^2 + \delta\|\boldsymbol{w}\|_1, \tag{2}$$

with $\|\cdot\|_1$ denoting the ℓ_1-norm and $\delta > 0$ being the regularization parameter. LASSO problems, also known as basis pursuit denoising, have been intensively investigated in the literature in the context of compressed sensing and sparse signal reconstruction (see, e.g., [19–21]). Despite non-smoothness of the LASSO objective function (2), it remains convex, which permits using a number of efficient estimation algorithms for estimating $\boldsymbol{w}$.

2.2. Distributed Signal Models

The optimization problem (2) can naturally be solved at a single fusion center that collects the measurement data from the agents in the network. Instead, the agents can perform this optimization in a cooperative manner by sharing their computational resources. To this end, two computing architectures can be considered. The first is termed the SOE approach, also known as homogeneous learning [9,22]. In SOE, the agents collect "private" input-output data pairs—examples or instances—and then cooperatively estimate a common vector $\boldsymbol{w}$, cf. Figure 1a.

Alternatively, each agent can infer its own model of the observed phenomenon, which are then be combined to form a global estimator, cf. Figure 1b. This is referred to as SOF or heterogeneous learning [8,9,23], since each agent has access to its own set of features, which are not known to the other agents. The agents, however, need to share the measurement data. In the following, we will consider both approaches to the distributed inference and exploration task.

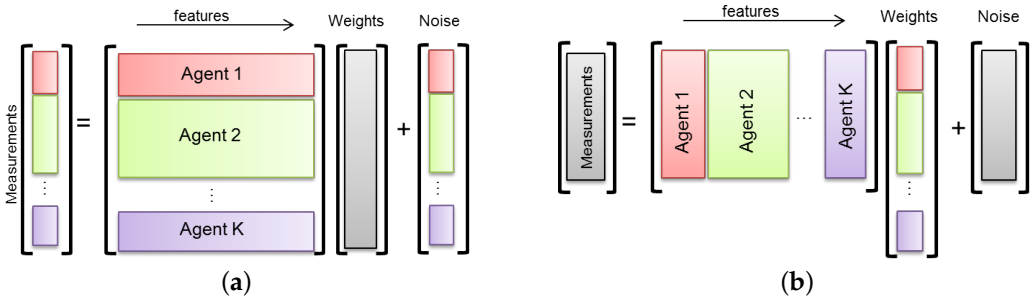

Figure 1. The colored areas represent an individual agent, and grey areas are commonly known by all agents. The matrices in these figures are arranged according to (1). (**a**) SOE learning approach: the weight vector $\boldsymbol{w}$ is commonly known by all agents; (**b**) SOF learning approach: the measurement vector $\boldsymbol{y}$ and noise vector $\boldsymbol{\epsilon}$ are commonly known by all agents.

2.2.1. Splitting-over-Examples

In SOE each agent k makes its own M_k local measurements $\{y[m], \boldsymbol{x}[m]\}$, $m = 1, \ldots, M_k$. The measurements of the k-th agent can therefore be arranged as $\boldsymbol{y}_k = [y_k[1], \ldots, y_k[M_k]]^T \in \mathbb{R}^{M_k}$ and $\boldsymbol{X}_k = [\boldsymbol{x}_k[1], \ldots, \boldsymbol{x}_k[M_k]]^T$. Hence, each agent k has a part of the global measurements as $\boldsymbol{y} = [y_1^T, \ldots, y_K^T]^T$ and $\boldsymbol{X}_\mathrm{M} = [\boldsymbol{X}_1^T, \ldots, \boldsymbol{X}_K^T]^T$. The local number of measurements M_k is related to the global measurements as $M = \sum_{k=1}^{K} M_k$. Prior to the estimation and data collection, the agents need to agree on the selection of the points $\boldsymbol{X}_\mathrm{N}$, features or support points that effectively define the discretization of the exploration environment $\mathcal{X}$. The number of these points N has to be selected large enough to ensure that the coverage of the exploration environment is sufficient. This number is a key trade-off parameter between efficiency and computational complexity of the resulting inference and exploration procedure, as we will show later.

Using local measurements, agent k defines its design matrix $\boldsymbol{\Psi}_k = \boldsymbol{\Psi}(\boldsymbol{X}_k, \boldsymbol{X}_\mathrm{N}) \in \mathbb{R}^{M_k \times N}$. Correspondingly, in SOE formulation, the model (1) is transformed as follows:

$$\boldsymbol{y}_k = \boldsymbol{\Psi}_k \boldsymbol{w} + \boldsymbol{\epsilon}_k, \; \forall \, k = 1, \ldots, K, \tag{3}$$

with $\boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \lambda^{-1} \boldsymbol{I})$. Note that within the SOE formulation, agents can experience different perturbations in $\boldsymbol{\epsilon}_k$.

Using (3), we can now recast (2) into a distributed optimization problem as:

$$\underset{\boldsymbol{w}}{\text{minimize}} \sum_{k=1}^{K} \frac{\lambda}{2} \left\| \boldsymbol{\Psi}_k \boldsymbol{w} - \boldsymbol{y}_k \right\|^2 + \delta \left\| \boldsymbol{w} \right\|_1 \tag{4}$$

To use ADMM, we introduce new local variables $\boldsymbol{z}_k \in \mathbb{R}^N \; \forall k = 1, \ldots, K$ and rewrite (4) as:

$$\begin{aligned} \underset{\boldsymbol{z}_k, \boldsymbol{w}}{\text{minimize}} \quad & \sum_{k=1}^{K} \frac{\lambda}{2} \left\| \boldsymbol{\Psi}_k \boldsymbol{z}_k - \boldsymbol{y}_k \right\|^2 + \delta \left\| \boldsymbol{w} \right\|_1 \\ \text{subject to} \quad & \boldsymbol{w} = \boldsymbol{z}_k \; \forall \, k = 1, \ldots, K. \end{aligned} \tag{5}$$

Note that the local variables z_k play the role of weights w in the LASSO objective function. In contrast to (2), however, the additional constraint in (5) ensures that the local weight estimates of z_k will eventually convergence to the same fixed point. The optimization (5) is now suited for ADMM [9]. In the following, we briefly outline the ADMM algorithm for solving (5) and refer the interested reader to [9] for more details.

To define the objective function, ADMM uses an augmented Lagrangian such that:

$$\mathcal{L}_{\mathrm{SOE}}(w, z_1, \ldots, z_K, u_1, \ldots, u_K) = \sum_{k=1}^{K} \frac{\lambda}{2} \|\Psi_k z_k - y_k\|^2 + \delta \|w\|_1 + u_k^T (w - z_k) + \frac{\rho}{2} \|w - z_k\|_2^2, \quad (6)$$

with $u_k \in \mathbb{R}^N$ being the dual variable and $\rho > 0$ defining the ADMM penalty parameter. The minimizer of the augmented Lagrangian (6) with respect to w, z_k and u_k, $k = 1, \ldots, K$ will be a solution to (5) [9]. The latter is found by sequentially minimizing $\mathcal{L}_{\mathrm{SOE}}$ first over z_k and w, followed by an update of the dual variable u_k [9]. By combining the linear and quadratic terms within the augmented Lagrangian in (6), a scaled form of the ADMM can be constructed. The corresponding algorithm is shown below.

1. At iteration $i = 0$, initialize $z_k^{[0]} = 0$, $w^{[0]} = 0$ and $u^{[0]} = 0$.
2. Compute an estimate $z_k^{[i+1]}$ by solving:

$$z_k^{[i+1]} = \arg\min_{z_k} \frac{\lambda}{2} \|\Psi_k z_k - y_k\|^2 + \frac{\rho}{2} \left\| z_k - w^{[i]} + u_k^{[i]} \right\|_2^2. \quad (7)$$

3. Use the averaged consensus algorithm of the network of agents for:

$$\bar{z}^{[i+1]} = \frac{1}{K} \sum_{k=1}^{K} z_k^{[i+1]}. \quad (8)$$

4. Compute an estimate $w^{[i+1]}$ of the weight vector by solving:

$$w^{[i+1]} = \arg\min_{w} \delta \|w\|_1 + \frac{K\rho}{2} \left\| w - \bar{z}^{[i+1]} - u^{[i]} \right\|_2^2. \quad (9)$$

5. Update the dual variable:

$$u_k^{[i+1]} = u_k^{[i]} + \bar{z}_k^{[i+1]} - w^{[i+1]}. \quad (10)$$

6. The algorithm is converged if $\|u^{[i+1]} - u^{[i]}\| < t$. Otherwise, i is incremented, and the algorithm continues at Step 2.

Let us point out that the communication step between the agents is only needed in (8) to compute the averaged vector $\bar{z}_k^{[i+1]}$. After the consensus update in (8), the other steps are calculated locally by each agent, yet an estimate of the weight vector $w^{[i]}$ will be the same for all agents by virtue of the ADMM algorithm.

2.2.2. Splitting-over-Features

In SOF, each agent infers a local model of the observed phenomenon, while the measurement data are shared among the agents. In our case, this implies that each agent defines its own separate set of N_k features, which might not necessarily be the same for the agents, and N_k is related to the global number of features as $N = \sum_{k=1}^{K} N_k$. Here, we will assume that each agent's features are defined at X_{N_k}, where an agent k has made a measurement, i.e., $X_{N_k} = X_{M_k} = [x_k[1], \ldots, x_k[M_k]]$. Then, the local design matrix in the SOF case can be defined as $\Phi_k = \Psi(X_{\mathrm{M}}, X_{N_k}) \in \mathbb{R}^{M \times N_k}$, where Φ_k is used for

a column-splitting of $\boldsymbol{\Psi}$ to better distinguish SOF with SOE. Consequently, the central measurement model in (1) can be reformulated in a distributed way as:

$$\boldsymbol{y} = \boldsymbol{\Phi}_1 \boldsymbol{w}_1 + \cdots + \boldsymbol{\Phi}_K \boldsymbol{w}_K + \boldsymbol{\epsilon} = \sum_{k=1}^{K} \boldsymbol{\Phi}_k \boldsymbol{w}_k + \boldsymbol{\epsilon}. \tag{11}$$

Note that in contrast to SOE, here, all agents naturally experience the same measurement noise. By inserting (11) into (2), we get a distributed LASSO as:

$$\underset{\boldsymbol{w}_k}{\text{minimize}} \qquad \frac{\lambda}{2} \left\| \sum_{k=1}^{K} \boldsymbol{\Phi}_k \boldsymbol{w}_k - \boldsymbol{y} \right\|^2 + \delta \sum_{k=1}^{K} \|\boldsymbol{w}_k\|_1 . \tag{12}$$

Again, we are aiming for a problem formulation, which is suited for a solution with ADMM. Thus, we define latent variables $\boldsymbol{v}_k = \boldsymbol{\Phi}_k \boldsymbol{w}_k$, $k = 1, \ldots, K$, that capture the response of individual agents and rewrite (12) as:

$$\begin{aligned} \underset{\boldsymbol{w}_k, \boldsymbol{v}_k}{\text{minimize}} \quad & \frac{\lambda}{2} \left\| \sum_{k=1}^{K} \boldsymbol{v}_k - \boldsymbol{y} \right\|^2 + \delta \sum_{k=1}^{K} \|\boldsymbol{w}_k\|_1 \\ \text{subject to} \quad & \boldsymbol{\Phi}_k \boldsymbol{w}_k = \boldsymbol{v}_k \; \forall \, k = 1, \ldots, K. \end{aligned} \tag{13}$$

We likewise solve the optimization (13) by using the ADMM algorithm [9]. In this case, the augmented Lagrangian for (13) can be written as:

$$\mathcal{L}_{\text{SOF}}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_K, \boldsymbol{s}) = \frac{\lambda}{2} \left\| \sum_{k=1}^{K} \boldsymbol{v}_k - \boldsymbol{y} \right\|^2 + \delta \sum_{k=1}^{K} \|\boldsymbol{w}_k\| + \frac{K\rho}{2} \sum_{k=1}^{K} \|\boldsymbol{\Phi}_k \boldsymbol{w}_k - \boldsymbol{v}_k + \boldsymbol{s}\|_2^2 , \tag{14}$$

where $\boldsymbol{s} \in \mathbb{R}^M$ is a dual variable of the ADMM algorithm. Now, using again the scaled version of (14) (see [9] Section 3.1 for more details), we minimize $\mathcal{L}_{\text{SOF}}$ using the following algorithm.

1. At iteration $i = 0$, initialize $\boldsymbol{w}_k^{[0]} = 0$, $\boldsymbol{s}^{[0]} = 0$ and $\boldsymbol{v}^{[0]} = 0$.
2. Compute an estimate $\boldsymbol{w}_k^{[i+1]}$ by solving:

$$\boldsymbol{w}_k^{[i+1]} = \arg\min_{\boldsymbol{w}_k} \left(\delta \|\boldsymbol{w}_k\|_1 + \frac{\rho}{2} \left\| \boldsymbol{\Phi}_k \boldsymbol{w}_k - \boldsymbol{\Phi}_k \boldsymbol{w}_k^{[i]} - \boldsymbol{v}^{[i]} + \overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i]} + \boldsymbol{s}^{[i]} \right\|_2^2 \right). \tag{15}$$

3. Use the averaged consensus algorithm of the network of agents to compute:

$$\overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i+1]} = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\Phi}_k \boldsymbol{w}_k^{[i+1]}. \tag{16}$$

4. Compute an estimate of the latent variable $\boldsymbol{v}^{[i+1]}$ of the weight vector by solving:

$$\boldsymbol{v}^{[i+1]} = (\lambda K + \rho)^{-1} \left(\lambda \boldsymbol{y} + \rho \left(\overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i+1]} + \boldsymbol{s}^{[i]} \right) \right). \tag{17}$$

5. Update the dual variable:

$$\boldsymbol{s}^{[i+1]} = \boldsymbol{s}^{[i]} + \overline{\boldsymbol{\Phi}\boldsymbol{w}}^{[i+1]} - \boldsymbol{v}^{[i+1]}. \tag{18}$$

6. The algorithm converges if $\|\boldsymbol{s}^{[i+1]} - \boldsymbol{s}^{[i]}\| < t$ is below some threshold t. Otherwise, i is incremented, and the algorithm continues at Step 2.

Observe that apart from (16), the other optimization steps of the ADMM algorithm are computed locally by each agent. After the optimization in (15), an averaged consensus over the network of agents is computed, i.e., all agents essentially compute a cooperative averaged response $\overline{\boldsymbol{\Phi w}}$ with (16).

2.2.3. The SOE versus the SOF Approach

Both SOE and SOF formulations attempt to find a solution to the optimization (2) using different data distribution approaches. Both realizations with the ADMM algorithm show similar approaches to the optimization: there is a consensus step, where a certain quantity is cooperatively computed by a multi-agent system, followed by several local optimizations or calculations. It is this cooperative step that eventually determines the requirement on the communication bandwidth to implement the scheme. Thus, we summarize the sizes of the vectors, which have to be communicated, in Table 1.

Table 1. Differences between the distribution paradigms.

	SOE	SOF
Estimation	• Distribute $\boldsymbol{w} \in \mathcal{R}^N$ • Static number of features • More intuitive to implement	• Distribute $\boldsymbol{\Phi}_k \boldsymbol{w}_k \in \mathcal{R}^M$ and $\{\boldsymbol{x}_m, y_m\}$ for each new measurement • Permits pruning unused kernels • Estimation is faster
Exploration	• Estimate $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{c}(\check{\boldsymbol{x}}) \in \mathbb{R}^{N\times 1}$ and $b(\check{\boldsymbol{x}})$ by means of a consensus. Each agent communicates its part. • $\check{\boldsymbol{x}} \in \mathcal{X}_\mathcal{I}$ does not need to be communicated, because every agent knows the set of inactive features. • Measurements are locally needed; thus, also $\boldsymbol{x} \in \mathcal{X}$ does not need to be communicated. • Agents can place features only at previously defined positions and are restricted to a predefined kernel.	• Estimate $\boldsymbol{D} \in \mathbb{R}^{M\times M}$, $\boldsymbol{e}(\boldsymbol{x}) \in \mathbb{R}^{M\times 1}$ and $g(\boldsymbol{x})$ by means of a consensus. • The agents have to communicate the proposed positions X_k. • Agents can place features everywhere and can use any kernel.

In particular, we see that in the case of SOE, the agents have to agree on the variable $\overline{\boldsymbol{z}}^{[i+1]}$ in (8), which has dimensionality of N—the number of support points in $\boldsymbol{X}_N$—which defines the discretization of the exploration area $\mathcal{X}$. This can be large and potentially exceed the locally available number of measurements, i.e., $M_k \ll N$. Instead, in SOF, the consensus is performed over an M-dimensional vector $\overline{\boldsymbol{\Phi w}}^{[i+1]}$ in (16).

Furthermore, we see that the complexity of the local optimization problem (7) in SOE is essentially dictated by the matrix inversion in (7). Applying the inversion naively, the complexity is $\mathcal{O}(N^3)$, the dimensionality of the estimated vector. In [9], Section 4, the authors discussed general patterns and provided methods for the reduction of the complexity, e.g., the matrix inversion can be reduced to $\mathcal{O}(NM_k^2)$. Instead, the complexity of (15) in SOF is dictated by the dimensionality of $\boldsymbol{w}_k$, the number of features collected by each agent. Thus, when $M \ll N_k$, SOF can be significantly more efficient as compared to SOE. Additionally, the agents keep their model size N_k low, which is in SOF permitted due to the individual models. However, the SOF approach requires a more involved communication protocol that makes measurements globally available. Depending on a particular application, this may pose a challenge for a practical realization of a distributed system.

Generally speaking, the complexity of SOE is mainly driven by the number of features N; whereas, the complexity of SOF is determined by the number of measurements M.

3. Distributed Cooperative Exploration

In this section, our goal is to build upon the algorithms discussed in the previous section to derive optimal exploration strategies, i.e., optimize swarm trajectories to increase the efficiency of data collection. To this end, we adopt ideas from optimal experiment design [24] and cast them in the distributed swarm exploration setting.

In optimal experiment design, the goal is to select an experiment setting that optimizes (reduces) some measure of an estimator's variance. Translated in our context, this corresponds to selecting optimal measurement locations, which reduce the uncertainty of model parameters. There are a number of criteria for optimizing the uncertainty of parameter estimates [24]. In this paper, we use a so-called D-optimality criterion, which attempts to minimize the log-determinant of the estimator's covariance matrix. In [24], the author showed that this is equal to the minimization of the differential Shannon entropy. Therefore, we will refer to D-optimality and entropy interchangeably.

In a distributed setting, this criterion cannot be computed in a simple fashion, since the required information is shared among agents. In what follows, we therefore develop schemes for a distributed computation of the D-optimality criterion separately for the SOE and SOF distribution paradigms. Similarly to Section 2, we will begin with a centralized setting and then extend the developed concepts to SOE and SOF.

3.1. Centralized Exploration

We begin with the LASSO optimization (2). On the one hand, the presence of the ℓ_1 penalty leads to a sparse estimate of the parameter vector $\boldsymbol{w}$; on the other hand, this term makes the resulting objective function non-smooth in $\boldsymbol{w}$. This makes computation of the D-optimality difficult, because the estimator's covariance is then not defined. As a consequence, the D-optimality criterion cannot be computed. In order to circumvent this difficulty, we approximate the estimator's covariance using the following approach.

Let $\delta_i \triangleq 2\delta/|w_i|$, $i = 1, \ldots, N$. This allows us to rewrite (2) as:

$$\begin{aligned} \underset{w_k}{\text{minimize}} \quad & \frac{\lambda}{2}\left\|\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)\boldsymbol{w} - \boldsymbol{y}\right\|^2 + \frac{1}{2}\sum_{i=1}^{N}\delta_i|w_i|^2, \\ \text{subject to} \quad & \delta_i = 2\delta/|w_i|. \end{aligned} \tag{19}$$

It is easy to see that, for a fixed δ_i, the resulting objective function becomes smooth in $\boldsymbol{w}$.

Now, consider the state of the algorithm, when a sparse estimate $\widehat{\boldsymbol{w}}$ that solves (19) is computed. Thus, we can approximate the covariance of the estimator in (2) by using the covariance of the estimator in (19). To this end, we use the latter to compute the inverse of the second order derivative evaluated at $\widehat{\boldsymbol{w}}$ with $\widehat{\delta}_i = 2\delta/|\widehat{w}_i|$, which is the estimators' covariance. Thus, the second order derivative of (19) is $\boldsymbol{C} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi} + \widehat{\boldsymbol{\Delta}})^{-1}$, where $\widehat{\boldsymbol{\Delta}} = \operatorname{diag}\{\widehat{\delta}_1, \ldots, \widehat{\delta}_N\}$. Let us point out that since $\widehat{\boldsymbol{w}}$ is sparse, $\widehat{\delta}_i \to \infty$ when $\widehat{w}_i \to 0$. This allows us to partition the columns of $\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)$ and a diagonal matrix $\widehat{\boldsymbol{\Delta}}$ into two sub-matrices:

1. Active features $\boldsymbol{\Psi}_{\mathcal{A}} \triangleq \boldsymbol{\Psi}_{\mathcal{A}}(\boldsymbol{X}_M, \boldsymbol{X}_N)$, which collects the columns of $\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)$ that correspond to non-zero weight estimates and finite regularization coefficients $\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}$.
2. Inactive features $\boldsymbol{\Psi}_{\mathcal{I}} \triangleq \boldsymbol{\Psi}_{\mathcal{I}}(\boldsymbol{X}_M, \boldsymbol{X}_N)$, which are associated with zero elements in $\widehat{\boldsymbol{w}}$ and correspond to inactive regularization parameters $\widehat{\boldsymbol{\Delta}}_{\mathcal{I}} = \infty\boldsymbol{I}$.

Clearly, $\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N) = [\boldsymbol{\Psi}_{\mathcal{A}}, \boldsymbol{\Psi}_{\mathcal{I}}]$ and $\widehat{\boldsymbol{\Delta}} = \operatorname{diag}\{\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}, \widehat{\boldsymbol{\Delta}}_{\mathcal{I}}\}$ (with an appropriate permutation of columns in $\boldsymbol{\Psi}(\boldsymbol{X}_M, \boldsymbol{X}_N)$). Obviously, inactive features $\boldsymbol{\Psi}_{\mathcal{I}}$ do not contribute to the variance of the weight estimate: the corresponding entries in $\boldsymbol{C}$ tend to zero as $\boldsymbol{\Delta}_{\mathcal{I}} \to \infty\boldsymbol{I}$. The covariance of active elements can thus be approximated by $\boldsymbol{C}_{\mathcal{A}} \triangleq (\lambda\boldsymbol{\Psi}_{\mathcal{A}}^T\boldsymbol{\Psi}_{\mathcal{A}} + \widehat{\boldsymbol{\Delta}}_{\mathcal{A}})^{-1}$, i.e., using the columns of the active components only.

Now, consider a potential measurement location $\boldsymbol{x} \in \mathcal{X}$, and let us study how $\boldsymbol{C}_{\mathcal{A}}$ changes when this measurement is to be taken into account. To simplify the notation, we neglect now the dependency of $\boldsymbol{\Psi}$ on $\boldsymbol{X}_M$ and $\boldsymbol{X}_N$. The new measurement will result in an "addition" of a new row vector to the design matrix $\boldsymbol{\Psi}$. For the active columns, we will denote this row vector as $\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{X}_N) = [\varphi(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \varphi(\boldsymbol{x}, \boldsymbol{x}_N)]^T_{\mathcal{A}}$, where subscript $\mathcal{A}$ indicates the restriction on N columns to the active subset. Additionally, a new measurement can potentially "activate" a feature, i.e., it might lead to some element in $\boldsymbol{w}$ becoming non-zero at the next estimation stage of the algorithm (it is reasonable to assume that at most one feature is activated per measurement as the number of degrees of freedom should not grow faster than the number of measurements). An activated feature will result in an extension of $\boldsymbol{\Psi}_{\mathcal{A}}$ with a new column vector $\boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}) = [\varphi(\boldsymbol{X}_1, \check{\boldsymbol{x}}), \ldots, \varphi(\boldsymbol{X}_M, \check{\boldsymbol{x}})]^T$, with $\check{\boldsymbol{x}} \in \mathcal{X}_{\mathcal{I}}$, where $\mathcal{X}_{\mathcal{I}}$ denotes the set of possible inactive feature locations.

Finally, due to the measurement at a location $\boldsymbol{x} \in \mathcal{X}$ a new design matrix $\widetilde{\boldsymbol{\Psi}}_{\mathcal{A}}(\boldsymbol{x}, \check{\boldsymbol{x}})$ can be constructed as follows:

$$\widetilde{\boldsymbol{\Psi}}_{\mathcal{A}}(\boldsymbol{x}, \check{\boldsymbol{x}}) = \begin{bmatrix} \boldsymbol{\Psi}_{\mathcal{A}} & \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}) \\ \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{X}_N)^T & \varphi(\boldsymbol{x}, \check{\boldsymbol{x}}) \end{bmatrix}, \tag{20}$$

and the corresponding "new" covariance is then given as $\widetilde{\boldsymbol{C}} = \left(\lambda \widetilde{\boldsymbol{\Psi}}_{\mathcal{A}}(\widetilde{\boldsymbol{x}}, \check{\boldsymbol{x}})^T \widetilde{\boldsymbol{\Psi}}_{\mathcal{A}}(\widetilde{\boldsymbol{x}}, \check{\boldsymbol{x}}) + \operatorname{diag}\{\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}, 0\}\right)^{-1}$. Here, setting the regularization for the new feature weight to zero puts no regularization on a potentially new placed feature. If the regularization is too high, we might place features at already measured positions, cf. Section 4.1. The D-optimality criterion yields a measurement location $\widehat{\boldsymbol{x}}$ that minimizes $\log\det \widetilde{\boldsymbol{C}}$, i.e.,

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathcal{X}, \check{\boldsymbol{x}} \in \mathcal{X}_{\mathcal{I}}}{\arg\min} \log \left| \lambda \widetilde{\boldsymbol{\Psi}}_{\mathcal{A}}(\boldsymbol{x}, \check{\boldsymbol{x}})^T \widetilde{\boldsymbol{\Psi}}_{\mathcal{A}}(\boldsymbol{x}, \check{\boldsymbol{x}}) + \begin{bmatrix} \widehat{\boldsymbol{\Delta}}_{\mathcal{A}} & \\ & 0 \end{bmatrix} \right|^{-1}. \tag{21}$$

We then proceed by inserting (20) into (21) and simplifying the result, which leads to:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathcal{X}, \check{\boldsymbol{x}} \in \mathcal{X}_{\mathcal{I}}}{\arg\max} \log \left| \begin{bmatrix} \lambda \boldsymbol{\Psi}_{\mathcal{A}}^T \boldsymbol{\Psi}_{\mathcal{A}} + \widehat{\boldsymbol{\Delta}}_{\mathcal{A}} & \lambda \boldsymbol{\Psi}_{\mathcal{A}}^T \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}) \\ \lambda \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}})^T \boldsymbol{\Psi}_{\mathcal{A}} & \lambda \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}})^T \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}) \end{bmatrix} + \lambda \begin{bmatrix} \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{X}_N) \\ \varphi(\boldsymbol{x}, \check{\boldsymbol{x}}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{X}_N) \\ \varphi(\boldsymbol{x}, \check{\boldsymbol{x}}) \end{bmatrix}^T \right|. \tag{22}$$

As we see, (22) must be evaluated for each potential measurement position $\boldsymbol{x}$ and for each potential feature position $\check{\boldsymbol{x}}$. This requires computing a determinant of a large matrix for every test position and a possible active feature, a computationally costly numerical problem. This computation can be implemented more efficiently by exploiting the structure of (22) as follows. Define now:

$$\boldsymbol{A} \triangleq \lambda \boldsymbol{\Psi}_{\mathcal{A}}^T \boldsymbol{\Psi}_{\mathcal{A}} + \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}, \quad \boldsymbol{c}(\check{\boldsymbol{x}})^T \triangleq \lambda \boldsymbol{\Psi}_{\mathcal{A}}^T \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}), \quad b(\check{\boldsymbol{x}}) \triangleq \lambda \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}})^T \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}). \tag{23}$$

With the definitions in (23), the cost function (22) can be re-cast as:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in \mathcal{X}, \check{\boldsymbol{x}} \in \mathcal{X}_{\mathcal{I}}}{\arg\max} \log \left| \begin{bmatrix} \boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{x}}) \\ \boldsymbol{c}(\check{\boldsymbol{x}})^T & b(\check{\boldsymbol{x}}) \end{bmatrix} + \lambda \begin{bmatrix} \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) \\ \varphi(\boldsymbol{x}, \check{\boldsymbol{x}}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})^T & \varphi(\boldsymbol{x}, \check{\boldsymbol{x}}) \end{bmatrix} \right|, \tag{24}$$

where we use $\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) = \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{X}_N)$ to lighten up the notation. We make use of the matrix determinant lemma to bring (24) in the following form:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{X},\check{\boldsymbol{x}}\in\mathcal{X}_\mathcal{I}}{\arg\max} \log\left|\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{x}})\\ \boldsymbol{c}(\check{\boldsymbol{x}})^T & b(\check{\boldsymbol{x}})\end{bmatrix}\right| + \log\left|1+\lambda\begin{bmatrix}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})\\ \varphi(\boldsymbol{x},\check{\boldsymbol{x}})\end{bmatrix}^T\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{x}})\\ \boldsymbol{c}(\check{\boldsymbol{x}})^T & b(\check{\boldsymbol{x}})\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})\\ \varphi(\boldsymbol{x},\check{\boldsymbol{x}})\end{bmatrix}\right|. \tag{25}$$

The first log-term in (25) can be shown to be equal to $\log\left|\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{x}})\\ \boldsymbol{c}(\check{\boldsymbol{x}})^T & b(\check{\boldsymbol{x}})\end{bmatrix}\right| = \log|\boldsymbol{A}| + \log|q(\check{\boldsymbol{x}})|$, where $q(\check{\boldsymbol{x}}) = b(\check{\boldsymbol{x}}) - \boldsymbol{c}(\check{\boldsymbol{x}})^T\boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{x}})$ is the Schur complement of $b(\check{\boldsymbol{x}})$ in the corresponding matrix under the logarithm. To simplify the second log-term in (25), we note that:

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{x}})\\ \boldsymbol{c}(\check{\boldsymbol{x}})^T & b(\check{\boldsymbol{x}})\end{bmatrix}^{-1} = \begin{bmatrix}\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{x}})q(\check{\boldsymbol{x}})^{-1}\boldsymbol{c}(\check{\boldsymbol{x}})^T\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{c}(\check{\boldsymbol{x}})/q(\check{\boldsymbol{x}})\\ -\boldsymbol{c}(\check{\boldsymbol{x}})^T\boldsymbol{A}^{-1}/q(\check{\boldsymbol{x}}) & 1/q(\check{\boldsymbol{x}})\end{bmatrix}. \tag{26}$$

Then,

$$\begin{aligned}&\log\left|1+\lambda\begin{bmatrix}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})\\ \varphi(\boldsymbol{x},\check{\boldsymbol{x}})\end{bmatrix}^T\begin{bmatrix}\boldsymbol{A} & \boldsymbol{c}(\check{\boldsymbol{x}})\\ \boldsymbol{c}(\check{\boldsymbol{x}})^T & b(\check{\boldsymbol{x}})\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})\\ \varphi(\boldsymbol{x},\check{\boldsymbol{x}})\end{bmatrix}\right|\\ &= \log\left(1+\lambda\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x}) + \lambda\left(\varphi(\boldsymbol{x},\check{\boldsymbol{x}}) - \boldsymbol{c}(\check{\boldsymbol{x}})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})\right)^2 q(\check{\boldsymbol{x}})^{-1}\right),\end{aligned} \tag{27}$$

and finally, by neglecting $\log|\boldsymbol{A}|$ due to its independence of $\check{\boldsymbol{x}}$ or $\boldsymbol{x}$, (24) can be shown as equivalent to:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathcal{X},\check{\boldsymbol{x}}\in\mathcal{X}_\mathcal{I}}{\arg\max} \log|q(\check{\boldsymbol{x}})| + \log\left(1+\lambda\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x}) + \lambda\left(\varphi(\boldsymbol{x},\check{\boldsymbol{x}}) - \boldsymbol{c}(\check{\boldsymbol{x}})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})\right)^2 q(\check{\boldsymbol{x}})^{-1}\right). \tag{28}$$

Expression (28) is a key to the proposed exploration that uses the D-optimality criterion for defining the next possible measurement location. Clearly, in a multi-agent setting, (28) can only be computed using cooperation as the required quantities of the covariance matrix are distributed among the agents. In the following, we discuss how this cooperation can be realized in SOE and SOF settings using averaged consensus.

3.2. Exploration for Splitting-over-Examples

Consider now an arbitrary agent k that attempts to find a next measurement position $\boldsymbol{x}_k \in \mathcal{X}_k \subseteq \mathcal{X}$ by solving (28).

To compute (28), we will distinguish between local quantities, which are only known to an individual agent and therefore do not require cooperation with other agents, and those that are distributed among the agents in the network. In particular, a vector $\boldsymbol{\psi}_\mathcal{A}(\boldsymbol{x})$ can be evaluated locally by each agent; similarly, computing $\varphi(\boldsymbol{x},\check{\boldsymbol{x}})$, as well as $\widehat{\boldsymbol{\Delta}}_\mathcal{A}$ does not require cooperation in the SOE approach.

In contrast, the quantities $\boldsymbol{A}$, $\boldsymbol{c}(\check{\boldsymbol{x}})$ and $b(\check{\boldsymbol{x}})$ in (28) are only available in a distributed form. Luckily, in SOE, the distributed quantities can be recovered by means of an averaged consensus algorithm [25,26]. Indeed, it can be recognized that:

$$\boldsymbol{A} \triangleq \lambda \boldsymbol{\Psi}_{\mathcal{A}}^T \boldsymbol{\Psi}_{\mathcal{A}} + \widehat{\boldsymbol{\Delta}}_{\mathcal{A}} = \sum_{k=1}^{K} \lambda \boldsymbol{\Psi}_{\mathcal{A},k}^T \boldsymbol{\Psi}_{\mathcal{A},k} + \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}, \tag{29}$$

$$\boldsymbol{c}(\check{\boldsymbol{x}}) \triangleq \lambda \boldsymbol{\Psi}_{\mathcal{A}}^T \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}) = \sum_{k=1}^{K} \lambda \boldsymbol{\Psi}_{\mathcal{A},k}^T \boldsymbol{\varphi}_k(\boldsymbol{X}_{M_k}, \check{\boldsymbol{x}}), \tag{30}$$

$$b(\check{\boldsymbol{x}}) \triangleq \lambda \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}})^T \boldsymbol{\varphi}(\boldsymbol{X}_M, \check{\boldsymbol{x}}) = \sum_{k=1}^{K} \lambda \boldsymbol{\varphi}_k(\boldsymbol{X}_{M_k}, \check{\boldsymbol{x}})^T \boldsymbol{\varphi}_k(\boldsymbol{X}_{M_k}, \check{\boldsymbol{x}}). \tag{31}$$

The sums over the agents in Equations (29)–(31) can be efficiently computed using averaged consensus and thus become available at each agent. Let us stress that Equations (29)–(31) depend on an unknown feature location $\check{\boldsymbol{x}}$. However, in SOE, the features are fixed a-priori, and every agent knows $\mathcal{X}_{\mathcal{I}}$. As such, each agent can "scan" $\check{\boldsymbol{x}} \in \mathcal{X}_{\mathcal{I}}$ without the participation of the other agents. This implies that solving (28) with respect to $\check{\boldsymbol{x}}$, Equations (29)–(31) can be computed for each element in $\mathcal{X}_{\mathcal{I}}$. Furthermore, as can be seen in Equations (29)–(31), the consensus parts do not depend on $\boldsymbol{x} \in \mathcal{X}$. Thus, the proposed measurement positions do not need to be distributed. Now, an agent k can evaluate (28) for any $\boldsymbol{x} \in \mathcal{X}_k$. The whole procedure is summarized in Algorithm 1.

Algorithm 1 Exploration for K agents with SOE.

procedure EXPLORATION FOR SOE
  for all $k = 1, \ldots, K$ **do**
    $\boldsymbol{A} \leftarrow$(29)
    $\widehat{\boldsymbol{x}}_k = \boldsymbol{x}_{M_K}$
    for all $\check{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{I}}$ **do**
      $\boldsymbol{c}(\check{\boldsymbol{x}}) \leftarrow$(30)
      $\boldsymbol{b}(\check{\boldsymbol{x}}) \leftarrow$(31)
    end for
    $\widehat{\boldsymbol{x}}_k \leftarrow$ (28), with $\boldsymbol{x} \in \mathcal{X}_k$
    Agent k moves and makes the measurement
  end for
end procedure

Regarding the communication, we evaluated the amount of floats that have to be communicated. As an initial communication, the agents estimate $\boldsymbol{A}$, where initially, $\mathcal{O}(|\mathcal{A}|^2)$ for each agent are sent to all neighbors. Due to the sparsity constraints, $|\mathcal{A}|$ will be typically small. During the evaluation of the exploration criterion, $N+1$ floats for each $\check{\boldsymbol{x}} \in \mathcal{X}_{\mathcal{I}}$ for each neighbor have to be communicated. In total, the communication per measurement location estimation per agent yields $\mathcal{O}\left(\left(|\mathcal{I}|(N+1) + |\mathcal{A}|^2\right)(K-1)\right)$. Indeed, this is just the amount of floats, which have to be sent; by using an efficient communication protocol and coding scheme, the amount of transmitted bits can be reduced further.

3.3. Exploration for Splitting-over-Features

The SOF differs from the SOE case in several respects that have an impact on computing (28). First, since each agent is able to construct its own model, there is no need to keep track of inactive features; in fact, in SOF, only the active features are kept. Despite this, we will still keep the subscript notation $\mathcal{A}$ for consistency with the SOE case. Another pleasing aspect of SOF is that the selection of possible features $\check{\boldsymbol{x}}$ is a completely local task solved by an individual agent: an agent decides independently which new feature will be added to the model. Therefore, here, we decide to place features at the selected measurement location, i.e., we let $\check{\boldsymbol{x}} = \boldsymbol{x}$. This additionally simplifies the optimization (28).

A careful analysis of (28) shows that in the SOF case, except for $\varphi(\boldsymbol{x},\boldsymbol{x})$, $\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})$ and $b(\boldsymbol{x})$, the values $\boldsymbol{A}$, $\boldsymbol{c}(\boldsymbol{x})$ and $\psi_{\mathcal{A}}(\boldsymbol{x})$ are distributed quantities in SOF. In contrast to SOE, they cannot be computed directly via consensus. Instead, we will show how the products $\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})$, $\boldsymbol{c}_{\mathcal{A}}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})$ and $q(\boldsymbol{x})$ can be efficiently computed cooperatively.

To this end, consider $\boldsymbol{A}$ in (23). Using the matrix inversion lemma, we can compute:

$$\boldsymbol{A}^{-1} = (\lambda\boldsymbol{\Psi}_{\mathcal{A}}^T\boldsymbol{\Psi}_{\mathcal{A}} + \widehat{\boldsymbol{\Delta}}_{\mathcal{A}})^{-1} = \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1} - \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T(\lambda^{-1}\boldsymbol{I} + \boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T)^{-1}\boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}. \tag{32}$$

The quantity $\boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T$ can be computed cooperatively. Indeed,

$$\boldsymbol{D} \triangleq \boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T = \sum_{k=1}^{K}\boldsymbol{\Psi}_{\mathcal{A},k}\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}^{-1}\boldsymbol{\Psi}_{\mathcal{A},k}^T, \tag{33}$$

where $\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k} = \operatorname{diag}\{\widehat{\delta}_{1,k},\ldots,\widehat{\delta}_{N_k,k}\}$ are the regularization parameters of the local weight vector $\boldsymbol{w}_k$. As we see, (33) can be computed in a distributed fashion using the averaged consensus algorithm [25,26]. Let us point out that since in SOF, only active features are kept, $\widehat{\boldsymbol{w}}_k$ is not sparse, and thus, $\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}$ is always bounded, i.e., (32) is a stable inversion.

Now, we can compute the Schur complement $q(\boldsymbol{x}) = b(\boldsymbol{x}) - \boldsymbol{c}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{c}(\boldsymbol{x})$. Inserting $b(\boldsymbol{x})$ and $\boldsymbol{c}(\boldsymbol{x})$ from (23) and $\boldsymbol{A}^{-1}$ from (32) in the expression for $q(\boldsymbol{x})$, and using (33), it can be shown after some algebra that:

$$\begin{aligned} q(\boldsymbol{x}) = {} & \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})^T\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x}) \\ & - \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})^T\boldsymbol{\Psi}_{\mathcal{A}}\left(\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1} - \widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T(\lambda^{-1}\boldsymbol{I} + \boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T)^{-1}\boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\right)\lambda\boldsymbol{\Psi}_{\mathcal{A}}^T\boldsymbol{\varphi}(\boldsymbol{X}_M,\check{\boldsymbol{x}}) \\ = {} & \boldsymbol{\varphi}(\boldsymbol{x})^T\left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D}\right)^{-1}\boldsymbol{\varphi}(\boldsymbol{x}). \end{aligned} \tag{34}$$

In a similar fashion, we compute $\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})$. Using (32), we obtain:

$$\begin{aligned} \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) = {} & \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})^T\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) - \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})^T\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T\left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D}\right)^{-1}\boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) \\ = {} & g(\boldsymbol{x}) - \boldsymbol{e}(\boldsymbol{x})^T(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D})^{-1}\boldsymbol{e}(\boldsymbol{x}), \end{aligned} \tag{35}$$

where we defined:

$$\boldsymbol{e}(\boldsymbol{x}) \triangleq \boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) = \sum_{k=1}^{K}\boldsymbol{\Psi}_{\mathcal{A},k}\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}^{-1}\boldsymbol{\psi}_{\mathcal{A},k}(\boldsymbol{x}), \tag{36}$$

$$g(\boldsymbol{x}) \triangleq \boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})^T\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) = \sum_{k=1}^{K}\boldsymbol{\psi}_{\mathcal{A},k}(\boldsymbol{x})^T\widehat{\boldsymbol{\Delta}}_{\mathcal{A},k}^{-1}\boldsymbol{\psi}_{\mathcal{A},k}(\boldsymbol{x}). \tag{37}$$

Expressions (36) and (37) can also be computed using averaged consensus, which allows an agent k to compute (35) distributively.

Finally, we show how to cooperatively compute $\boldsymbol{c}_{\mathcal{A}}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x})$. Again, inserting $\boldsymbol{c}(\boldsymbol{x})$ from (23) and $\boldsymbol{A}^{-1}$ from (32) in the latter expression, we obtain:

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{x})^T\boldsymbol{A}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) = {} & \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})^T\boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) - \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})^T\boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\Psi}_{\mathcal{A}}^T\left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D}\right)^{-1}\boldsymbol{\Psi}_{\mathcal{A}}\widehat{\boldsymbol{\Delta}}_{\mathcal{A}}^{-1}\boldsymbol{\psi}_{\mathcal{A}}(\boldsymbol{x}) \\ = {} & \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})^T\boldsymbol{e}(\boldsymbol{x}) - \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})^T\boldsymbol{D}\left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D}\right)^{-1}\boldsymbol{e}(\boldsymbol{x}) \\ = {} & \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M,\boldsymbol{x})^T\left(\boldsymbol{I} - \boldsymbol{D}\left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D}\right)^{-1}\right)\boldsymbol{e}(\boldsymbol{x}), \end{aligned} \tag{38}$$

which can be computed distributively using (33) and (36).

Now, we can bring all ingredients together and reformulate (28) for the SOF scenario by using Equations (35) and (38) as follows:

$$\begin{aligned}\widehat{\boldsymbol{x}}_k = \underset{\boldsymbol{x}\in\mathcal{X}_k}{\arg\max}\ & \log \boldsymbol{\varphi}(\boldsymbol{x})^T \left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D}\right)^{-1} \boldsymbol{\varphi}(\boldsymbol{x}) \\ & + \log\left(1 + \lambda\left(g(\boldsymbol{x}) - \boldsymbol{e}(\boldsymbol{x})^T(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D})^{-1}\boldsymbol{e}(\boldsymbol{x})\right) + \right. \\ & \left. \frac{\lambda}{q(\boldsymbol{x})}\left(\varphi(\boldsymbol{x}) - \lambda\boldsymbol{\varphi}(\boldsymbol{X}_M, \boldsymbol{x})^T\left(\boldsymbol{I} - \boldsymbol{D}\left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{D}\right)^{-1}\right)\boldsymbol{e}(\boldsymbol{x})\right)^2\right), \end{aligned} \tag{39}$$

with quantities $\boldsymbol{D}$, $\boldsymbol{e}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ computed cooperatively with averaged consensus using (33), (36) and (37), respectively. Note that, for computing Equations (33), (36) and (37), each agent has to know $\boldsymbol{x} \in \mathcal{X}_k$. This involves an additional communication step, where agent k sends all possible measurement locations $\mathcal{X}_k$ to its neighbors. The resulting procedure is summarized in Algorithm 2.

Algorithm 2 Exploration for K agents with SOF.

procedure EXPLORATION FOR SOF
  for all $k = 1, \ldots, K$ **do**
    $\boldsymbol{D} \leftarrow$(33)
    Send $\mathcal{X}_k$ to all neighbors
    $\widehat{\boldsymbol{x}}_k = \boldsymbol{x}_{M_k}$, $l = -\infty$
    for all $\boldsymbol{x} \in \mathcal{X}_k$ **do**
      $\boldsymbol{e}(\boldsymbol{x}) \leftarrow$(36)
      $\boldsymbol{g}(\boldsymbol{x}) \leftarrow$(37)
    end for
    $\widehat{\boldsymbol{x}}_k \leftarrow$ (39)
    Agent k moves and makes the measurement
  end for
end procedure

The communication for the exploration criterion for SOF can also be evaluated with respect to the transmitted floats. Initially, the agents transmit M^2 floats to their neighbor to estimate $\boldsymbol{D}$. To evaluate the exploration criterion, agent k has to tell each other agent the considered measurement locations. Thus, agent k transmits $2|\mathcal{X}_k|$ in a two-dimensional environment to its neighbors. For the evaluation of the exploration criterion, each of agent k's neighbors transmits $M + 1$ for each $\boldsymbol{x} \in \mathcal{X}_k$ to agent k. In summary, $\mathcal{O}\left(\left(M^2 + |\mathcal{X}_k|(M+1)\right)(K-1) + 2|\mathcal{X}_k|\right)$ floats are transmitted for the estimation of agent k's next measurement location.

4. Simulations

In this section, we benchmark our proposed exploration schemes against other exploration strategies. We begin by considering the impact of the regularization introduced in Section 3.3 and then discuss the simulation parameters and the exploration results.

4.1. Impact of the Regularization for the Exploration Criterion

The empirical modification of the cost function in (19) makes the computation of the estimator's covariance matrix feasible. In the following, we study the impact of such a regularization on the used exploration criterion.

We consider now a simple one-dimensional scenario, with positions $\mathcal{X} \subset \mathbb{R}$ and two features; thus, $N = 2$. The features are modeled by a one-dimensional Gaussian with a fixed width σ and a feature position μ as:

$$\varphi(x, \mu) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Further, we define two measurement locations $x[1]$ and $x[2]$ for this scenario, and let $x[1] = -4$ and $x[2] = 4$. The two features are now defined as $\varphi_1(\boldsymbol{X}_M, x[1])$ and $\varphi_2(\boldsymbol{X}_M, x[2])$, and thus, $\boldsymbol{\Psi} = [\varphi_1(\boldsymbol{X}_M, x[1]), \varphi_2(\boldsymbol{X}_M, x[2])]$. The weights $\widehat{\boldsymbol{w}}$ of this scenario are then estimated by solving a centralized LASSO problem (2). Now, we will vary the value of the penalty parameter δ in the range $[0.001, \ldots, 500]$. The parameters $\widehat{\delta}_i = 2\delta/|\widehat{w}_i|$ are computed and inserted in the exploration objective $\log\det \widetilde{\boldsymbol{C}}^{-1}$ (cf. (28)). The corresponding results of this evaluation are plotted in Figure 2.

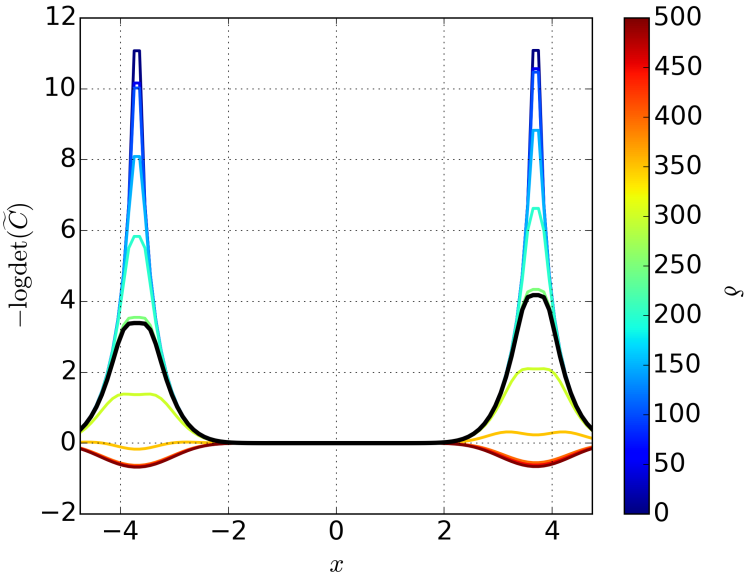

Figure 2. Variation of the regularization parameter δ and its influence on the exploration criterion. The black curve shows the case for $\delta = \lambda^{-2}\sqrt{2\log(M)}$, where $M = 2$.

As we see, whenever $\delta \to 0$, no regularization is used, and the LASSO objective function reduces to the standard least squares problem. In this case, the exploration criterion becomes highly peaked at existing measurement locations; the algorithm then selects measurement points away from the existing measurement location, as expected. When $\delta \to \infty$, the corresponding weight estimates naturally converge towards zero. In this case, the exploration criterion does not promote any new measurement at all: the value of the exploration objective $\log\det \widetilde{\boldsymbol{C}}^{-1}$ is minimized at existing measurements $x[1]$ and $x[2]$. As a result, over-regularization will force the agents to stay at current locations rather than explore the whole process.

Clearly, for a fixed δ, this effect depends on the estimated weights and on the used kernels. Therefore, to select the regularization parameter, we follow the strategy proposed in [27] Section 5.9, where the penalty parameter is adaptively chosen as $\delta = \lambda^{-2}\sqrt{2\log(M)}$. The corresponding value of the exploration criterion is shown in Figure 2 as a thick black line. Such a choice of δ has been found to be a good choice for achieving a stable regularization, while at the same time allowing agents to explore the field.

4.2. Evaluation of Entropy-Driven Exploration with Real Data

We are now ready to evaluate the proposed exploration strategies. As data, for exploration, we used the magnetic field intensity on the ground in our laboratory. These intensities have been measured with a magnetic field sensor module integrated in the Xsens MTx Package [14]. To evaluate the proposed entropy-driven trajectory planner, we created a simulation environment that uses a part of the magnetic field data of our laboratory. Figure 3 shows the data. The data are sampled on a grid at the positions $\boldsymbol{X} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{|\mathcal{X}'|}]^T$, where $\boldsymbol{x} \in \mathcal{X}' \subset \mathbb{R}^2$ and $|\mathcal{X}'| = 4128$. The magnetic field clearly shows a spatial correlation and therefore is a good starting point to evaluate the algorithm.

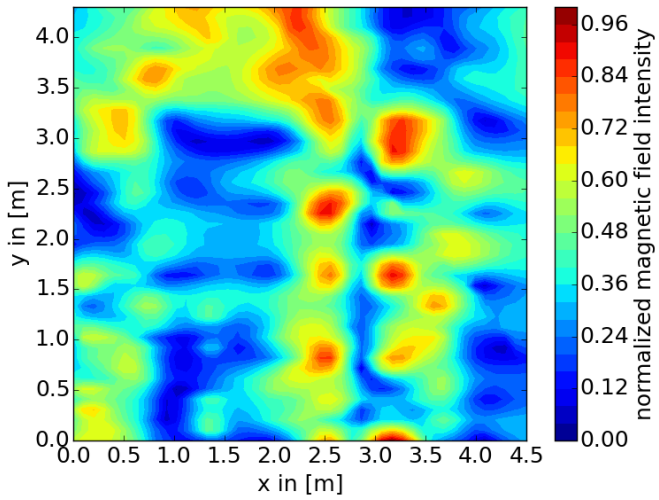

Figure 3. A part of the magnetic field on the ground of our laboratory. This plot shows the normalized magnetic field intensity in color coding. Red corresponds to a high magnetic field intensity, and blue corresponds to a low magnetic field intensity.

As features, we use a two-dimensional Gaussian kernel defined by a kernel width σ and a feature position $\boldsymbol{\mu} \in \mathbb{R}^2$ as:

$$\varphi(\boldsymbol{x}[i], \boldsymbol{\mu}) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}[i] - \boldsymbol{\mu}\|^2\right), \boldsymbol{x}[i] \in \mathcal{X}'.$$

In this paper, we set the feature width $\sigma = 30\,\text{cm}$ and the number of agents to $K = 10$. The agent's locations are initialized uniformly distributed over $\mathcal{X}'$. The predefined feature locations for SOE are all discretization points in $\mathcal{X}'$. The first measurements are then taken at their initial positions. We consider each agent in this simulation as a single point, and they can move everywhere within an instant. Therefore, we consider no collision avoidance in this simulation. To benchmark our exploration strategy, we will compare three different types of movement strategies: entropy, meander and random walk strategies. Entropy is the proposed strategy, whereas meander and random walk are used as benchmark algorithms. They are explained in the following.

Entropy: After the distributed estimation, the agents hierarchically evaluate the exploration criterion. For SOF, agent k distributes all possible measurement locations $\mathcal{X}_k$ to its neighbors $\mathcal{N}_k$ and receives each agent's consensus part, cf. Section 3.3. For SOE, the possible measurement locations do not need to be distributed, cf. Section 3.2. Then, for both paradigms, the exploration criterion is evaluated. Afterward, agent k chooses the location, which yields the best criterion according to (39) for SOF or (28) for SOE. Both schemes are also shown in Algorithms 1 and 2 for SOE and SOF, respectively.

Meander: In this exploration strategy, the agents start at an individual predefined position, somewhere at the left border of the environment, i.e., at coordinates with $\boldsymbol{x}_k[0] = [0, \frac{4.3}{k}]^T$, where 4.3 is the maximum of the environment in the y-direction. After estimation, they continue to move towards the opposing border on the right, i.e., they move in the x-direction with a predefined step size. In this

paper, the step size is equal to the kernel width σ. On the other side of the border, they move upwards and move back to the other side of the environment, generating a zigzag pattern.

Random walk: This is the simplest approach, where the agents choose a new measurement location by random uniform sampling of the whole $\mathcal{X}'$.

For meander and random walk, we do not evaluate the exploration criterion for the agents' movements. Consequently, the agents do not exchange the information of their already traversed paths, their proposed measurement positions $\mathcal{X}_k$, nor consensus parts for evaluation of the exploration criterion. For the entropy driven approach, Table 1 shows a detailed summary of the exchanged data.

For exploration with the SOF paradigm, we choose the ADMM parameter ρ according to [9] Section 3.4. There, the authors evaluate the primal and dual residuals and automatically choose ρ such that the difference between residuals is not too large, which is called residual balancing. However, for SOE, the residual balancing leads to an unstable solution of ρ, which might be related to the high number of features N and the comparably smaller number of measurements M. Therefore, we keep $\rho = 1.0$ as a constant for SOE, which has been evaluated by cross-validations (this empirical evaluation has to be done for each dataset, because the parameter ρ is highly data dependent) For both approaches, we set the parameter $\delta = \lambda^{-2}\sqrt{2\log(M)}$.

The estimations after 250 measurements of a single run for SOE are shown in the Figure 4a–c and for SOF in Figure 4d–f. The colored circles represent the measurement locations of the individual agents. The feature positions of the active features, i.e., the features with corresponding non-zero weight, are shown with black crosses. We observe that the entropy-driven methods tend to measure the border of the environments (cf. Figure 4a,d). This is a numerical effect of the evaluation of the entropy and has also been seen in [28]. The random walk has a uniformly distributed measurement location; see Figure 4b,e. It achieves a good coverage of the whole area. Therefore, active features might not be observed, and hence, the error is decreasing slower, compared to the entropy-driven method. The meander approach has always the drawback that it has to be carefully designed to achieve a good coverage within reasonable time. The trajectory design should reflect the observed process. However, this process is unknown from the beginning, and therefore, assumptions about measurement positions have to be made beforehand. This most often results in a bad performance or to many unuseful measurement positions. In summary, the entropy-driven methods have a better coverage, compared to the benchmark methods.

Now, we would like to discuss the number of active components in all approaches, i.e., the sparsity of the estimations. The number of active features is shown in Figure 5. The exploration strategy for SOF, cf. Figure 5b, leads to measuring most often relevant features. Therefore, features are only neglected in the model, when a sufficient number of measurements and features already exists. The benchmark methods lead to similar results, however.

SOE (see Figure 5a) has more fluctuations in the number of active components, because its number of features N is predefined from the beginning of the exploration. Therefore, in the beginning, the parameter $\delta = \lambda^{-2}\sqrt{2\log(M)}$ is suboptimal due to the bad ratio of M to N. However, when the number of measurements increase, active components are better identified, as can be seen in Figure 5a at 180 measurements.

Now, we compare the performance of the algorithm with respect to the NMSE of the estimated field. The metric is defined as:

$$e_{\text{NMSE, SOE}} = \frac{\|\boldsymbol{\Psi}(\boldsymbol{X}, \boldsymbol{X}_N)\boldsymbol{w} - \boldsymbol{f}\|}{\|\boldsymbol{f}\|}, \qquad e_{\text{NMSE, SOF}} = \frac{\left\|\sum_{k=1}^{K} \boldsymbol{\Psi}(\boldsymbol{X}, \boldsymbol{X}_{N_k})\boldsymbol{w}_k - \boldsymbol{f}\right\|}{\|\boldsymbol{f}\|}, \tag{40}$$

for SOE and SOF, respectively, where $\boldsymbol{f} = [f(\boldsymbol{x}_0), \ldots, f(\boldsymbol{x}_{|\mathcal{X}'|})]^T, \boldsymbol{x} \in \mathcal{X}'$ are all sampled values of the observed true magnetic field.

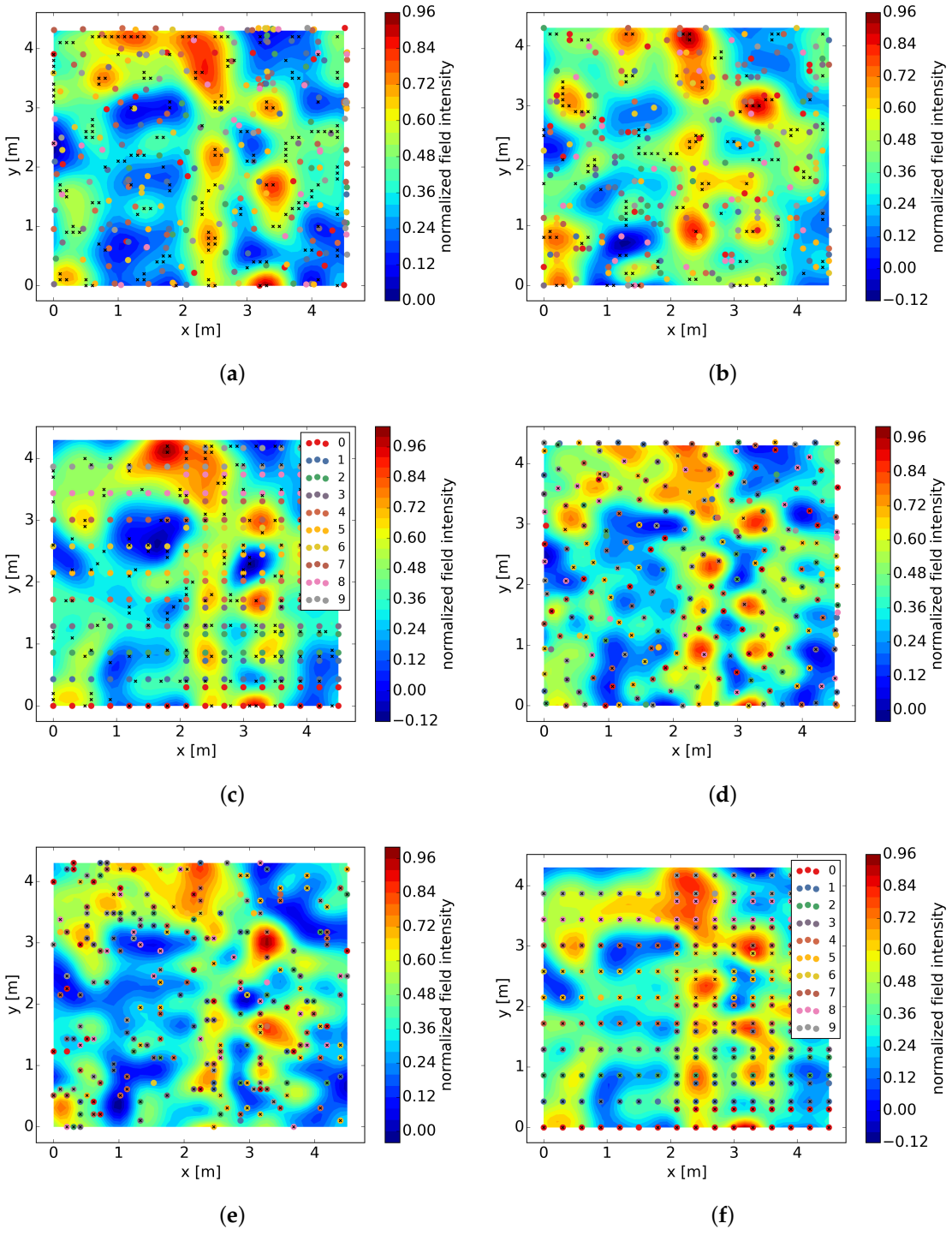

Figure 4. Estimation results for a single run of the estimation algorithm with the SOF and SOE splitting approaches. The measurement positions of the individual agents are marked with colored circles. The active features are marked by with black crosses. (**a**–**c**) show the estimations of an SOE approach, and (**d**–**e**) show the estimations of an SOF approach. (**a**) SOE estimation with entropy-driven exploration (**b**) SOE estimation with random exploration strategy; (**c**) SOE estimation with meander exploration strategy; (**d**) SOF estimation with entropy-driven exploration; (**e**) SOF estimation with random exploration strategy; (**f**) SOF estimation with meander exploration strategy.

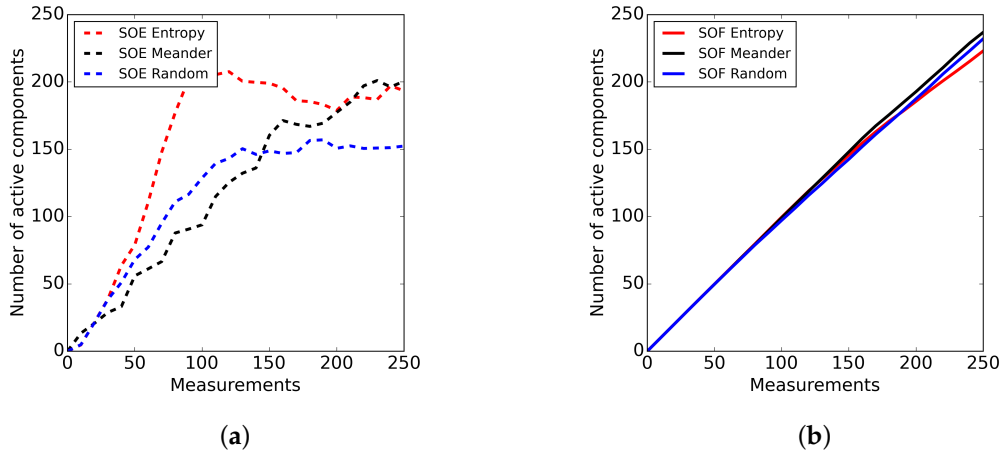

Figure 5. (**a**) Active components for SOE (dashed) for 10 agents averaged over 10 runs; (**b**) active components for SOF (solid) for 10 agents averaged over 10 runs.

The performance with respect to the NMSE is shown in Figure 6. Compared to the other methods, the entropy-driven exploration reduces the error faster with less measurements. However, the SOF and SOE splitting approaches have different performances. In this application, the measurements are few in the beginning of the exploration, and SOF clearly has an advantage due to its smaller number of features. As already mentioned, for the meander strategy, the underlying process influences the error with respect to the measurements. Therefore, the error drops faster only if active features have been measured. The random walk tends to measure at already measured or less beneficial locations and, thus, decreases more slowly, compared to the entropy approaches. Yet, compared to the meander, the random walk coverages faster and performs therefore better than the meander. In summary, the entropy-driven methods reduce the error faster than the benchmark methods.

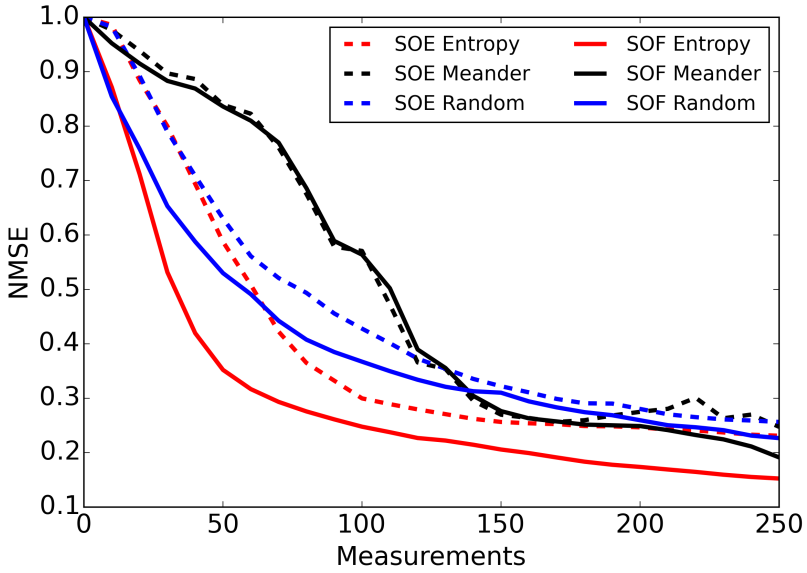

Figure 6. NMSE for SOE (dashed) and SOF (solid) for 10 agents, averaged over 10 runs.

5. Conclusions

In this paper, we discussed distributed learning and exploration algorithms for two different data distribution paradigms, namely SOE and SOF. A general linear model was used to describe a static spatial process that is explored with a set of multiple agents. In both, SOE and SOF, cases, we used the ADMM algorithm to estimate model parameters in a distributed fashion over the communication network between the agents. The used model also explicitly included an ℓ_1 penalty term that allowed computing a sparse parameter vector estimate. This penalty, on the one hand, regularizes the numerical optimization problem when only a few measurements are available; on the other hand, it leads to a sparse signal estimate, which allows reducing the model complexity at each agent. The latter allows reducing the communication load on the network (as in the SOE case) or reducing the local computational load per agent (as in the SOF case).

As an exploration objective, we proposed to use a D-optimality criterion from the theory of experiment design. This criterion is designed to select new measurement positions, which reduce the estimator's covariance. Yet, due to the use of sparsity constraints, the evaluation of the D-optimality criterion is difficult due to the non-smoothness of the resulting objective function. To circumvent this difficulty, we proposed a modification of the objective function that allows approximating the estimator's covariance. We then proposed algorithms that permit optimizing the resulting approximate D-optimality criterion in a distributed fashion using averaged consensus for both SOE and SOF data distribution paradigms.

Finally, we performed numerical simulations where the exploration criterion for each paradigm was tested against benchmark algorithms. These benchmark algorithms cover the types of systematic and random exploration strategies. Simulations have shown that the proposed exploration strategy outperforms the corresponding benchmarks with respect to the NMSE. We found that the SOF method has more flexibility with selecting feature positions and thus yields a lower NMSE compared to the SOE approach. It is possible that this effect can be attributed to the ratio between measurements M and features N. This ratio is very low for SOE in the beginning of the exploration, because the number of features is predefined and rather large, thus the difference in the NMSE performance. The SOF can, in contrast, build up the number of features in accordance with the number of measurements and can thus better fit the data. Regarding the number of measurements, we can also conclude that the proposed exploration strategy permits collecting relevant data more efficiently, as compared to the benchmark schemes, where the agents are not driven to more informative locations.

Both approaches seem very interesting for developing real distributed systems with efficient navigation algorithms and are next to be tested in real experiments. SOE, with a fixed number of features and a splitting in measurements, is useful when many measurements have to be observed, i.e., when we expect that $M \gg N$. For few features, this approach leads to fast computations. Furthermore, the communication with neighbors for computing the exploration criterion is reduced, since it is dependent on the number of features. The SOF scheme is computationally more efficient when the number of features is higher than the number of measurements, i.e., $N \gg M$. Therefore, it can be seen as an alternative to SOE in this case.

Author Contributions: Formal analysis, C.M. and D.S. Investigation, C.M. Software, C.M. Supervision, D.S. Writing, original draft, C.M. Writing, review and editing, C.M. and D.S.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NMSE	normalized mean square error
SAR	synthetic aperture radar
LASSO	least absolute shrinkage and selection operator
ADMM	alternating direction method of multipliers

SOE splitting-over-examples
SOF splitting-over-features
iid independent and identically distributed
SLAM simultaneous localization and mapping
GNSS global navigation satellite system

References

1. Seeni, A.; Schäfer, B.; Hirzinger, G. Robot Mobility Systems for Planetary Surface Exploration—State-of-the-Art and Future Outlook: A Literature Survey. In *Aerospace Technologies Advancements*; Thawar, T., Ed.; InTech: Rijeka, Croatia, 2010.
2. Avellar, G.; Pereira, G.; Pimenta, L.; Iscold, P. Multi-UAV Routing for Area Coverage and Remote Sensing with Minimum Time. *Sensors* **2015**, *15*, 27783–27803. [CrossRef] [PubMed]
3. Truszkowski, W.; Hinchey, M.; Rash, J.; Rouff, C. NASA's Swarm Missions: The Challenge of Building Autonomous Software. *IT Prof.* **2004**, *6*, 47–52. [CrossRef]
4. Galceran, E.; Carreras, M. A Survey on Coverage Path Planning for Robotics. *Robot. Auton. Syst.* **2013**, *61*, 1258–1276. [CrossRef]
5. Barrientos, A.; Colorado, J.; del Cerro, J.; Martinez, A.; Rossi, C.; Sanz, D.; Valente, J.A. Aerial Remote Sensing in Agriculture: A Practical Approach to Area Coverage and Path Planning for Fleets of Mini Aerial Robots. *J. Field Robot.* **2011**, *28*, 667–689. [CrossRef]
6. Maza, I.; Ollero, A. Multiple UAV Cooperative Searching Operation Using Polygon Area Decomposition and Efficient Coverage Algorithms. In *Distributed Autonomous Robotic Systems 6*; Alami, R., Chatila, R., Asama, H., Eds.; Springer: Tokyo, Japan, 2007; pp. 221–230.
7. Van Lon, R.R.S.; Holvoet, T. When Do Agents Outperform Centralized Algorithms?: A Systematic Empirical Evaluation in Logistics. *Auton. Agents Multi-Agent Syst.* **2017**, *31*, 1578–1609. [CrossRef]
8. Zheng, H.; Kulkarni, S.R.; Poor, H.V. Attribute-Distributed Learning: Models, Limits, and Algorithms. *IEEE Trans. Signal Process.* **2011**, *59*, 386–398. [CrossRef]
9. Boyd, S. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.* **2010**, *3*, 1–122. [CrossRef]
10. Côté, F.D.; Psaromiligkos, I.N.; Gross, W.J. In-Network Linear Regression with Arbitrarily Split Data Matrices. In Proceedings of the 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Washington, DC, USA, 7–9 December 2016; pp. 580–584.
11. Eich, M.; Hartanto, R.; Kasperski, S.; Natarajan, S.; Wollenberg, J. Towards Coordinated Multirobot Missions for Lunar Sample Collection in an Unknown Environment: Towards Coordinated Multi-Robot Missions for Lunar Sample Collection. *J. Field Robot.* **2014**, *31*, 35–74. [CrossRef]
12. Stachniss, C.; Grisetti, G.; Burgard, W. Information Gain-Based Exploration Using Rao-Blackwellized Particle Filters. In Proceedings of the Robotics: Science and Systems I. Robotics: Science and Systems Foundation, Cambridge, MA, USA, 8–11 June 2005.
13. Vallve, J.; Andrade-Cetto, J. Mobile Robot Exploration with Potential Information Fields. In Proceedings of the 2013 European Conference on Mobile Robots, Barcelona, Spain, 25–27 September 2013; pp. 222–227.
14. Viseras, A.; Wiedemann, T.; Manss, C.; Magel, L.; Mueller, J.; Shutin, D.; Merino, L. Decentralized Multi-Agent Exploration with Online-Learning of Gaussian Processes. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4222–4229.
15. Zhu, Z.; Tang, G.; Setlur, P.; Gogineni, S.; Wakin, M.B.; Rangaswamy, M. Super-Resolution in SAR Imaging: Analysis with the Atomic Norm. In Proceedings of the 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM), Rio de Janerio, Brazil, 10–13 July 2016; pp. 1–5.
16. Manss, C.; Wiedemann, T.; Shutin, D. Entropy Driven Height Profile Estimation with Multiple UAVs under Sparsity Constraints. In Proceedings of the 2017 IEEE Globecom Workshops (GC Wkshps), Singapore, 4–8 December 2017; pp. 1–6.
17. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*; The MIT Press: Cambridge, MA, USA, 2005.

18. Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *J. R. Stat. Soc. Ser. B (Methodol.)* **1996**, *58*, 267–288.
19. Donoho, D. Compressed Sensing. *IEEE Trans. Inf. Theory* **2006**, *52*, 1289–1306. [CrossRef]
20. Chen, S.; Donoho, D.; Saunders, M. Atomic Decomposition by Basis Pursuit. *SIAM Rev.* **2001**, *43*, 129–159. [CrossRef]
21. Baraniuk, R.G. Compressive Sensing [Lecture Notes]. *IEEE Signal Process. Mag.* **2007**, *24*, 118–121. [CrossRef]
22. Predd, J.B. Topics in Distributed Inference. Ph.D. Thesis, Princeton University, Princeton, NJ, USA, 2004.
23. Zheng, H.; Kulkarni, S.R.; Poor, H.V. Dimensionally Distributed Learning Models and Algorithm. In Proceedings of the 2008 11th International Conference on Information Fusion, Cologne, Germany, 30 June–3 July 2008; pp. 1–8.
24. Pukelsheim, F. *Optimal Design of Experiments*; SIAM: Philadelphia, PA, USA, 1993.
25. Nedic, A.; Ozdaglar, A.; Parrilo, P. Constrained Consensus and Optimization in Multi-Agent Networks. *IEEE Trans. Autom. Control* **2010**, *55*, 922–938. [CrossRef]
26. Olfati-Saber, R.; Fax, J.A.; Murray, R.M. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceed. IEEE* **2007**, *95*, 215–233. [CrossRef]
27. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed.; Springer Series in Statistics; Springer: New York, NY, USA, 2009.
28. Krause, A.; Singh, A.; Guestrin, C. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *J. Mach. Learn. Res.* **2008**, *9*, 235–284.