Distributed estimation of scalar fields with implicit coordination

Lorenzo Booth and Stefano Carpin*

Department of Computer Science and Engineering.
University of California, Merced
5200 N Lake Rd, Merced, CA 95343, USA,
lbooth@ucmerced.edu,scarpin@ucmerced.edu

Abstract. Motivated by our ongoing work in robotics for precision agriculture, in this work we consider the problem of estimating a scalar field using a team of robots collecting samples and subject to a travel budget. Our fully distributed method leverages the underlying properties of Gaussian Process regression to promote dispersion using minimal information sharing. Extensive simulations demonstrate that our proposed solution outperforms alternative approaches.

Keywords: Coordination; Applications in Agriculture; Estimation

1 Introduction

In this work we consider the problem of estimating a scalar field using a team of collaborating robots. This problem is motivated by our ongoing research in precision agriculture, where it is often necessary to estimate the spatial distribution of parameters such as soil moisture, nitrates, or carbon dioxide flux that can be modeled as spatially-varying scalars. To model this underlying field, we use a Gaussian process (GP) [6]. GPs are not only elegant and efficient computational tools to solve regression problems, but are also the model of choice in geostatistics and agroecological modeling, where GP regression is known as *kriging* [9]. A key feature about GP regression is that this model allows to easily estimate the uncertainty of the predictions, thus enabling iterative methods where new sensor measurements are collected in regions of high uncertainty to refine accuracy.

In agricultural applications one is often faced with the problem of estimating quantities over very large domains, therefore the use of multiple robots allows for quicker coverage of the region of interest. In these conditions, robots have to plan their motions with multiple objectives. When an exhaustive, systematic coverage

*L. Booth is supported by the Labor & Automation in California Agriculture (LACA) project (UC-MRPI intiative). S. Carpin is partially supported by the (NSF) under NSF Cooperative Agreement Number EEC-1941529 (IoT4Ag) and USDA/NIFA under award # 2021-67022-33452. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

of the entire region is not feasible, it is important to collect samples at the most informative places. It is also necessary to be aware of the limited distance that can be covered before the robot must be refueled or recharged. Therefore robots have to plan paths that will eventually terminate at a designated end point before they run out of energy or fuel. Likewise, it is also important to consider that in real world applications the energy consumed to move between two locations is not deterministically known upfront; rather, it is a random variable whose realization is only known at run-time. For example, a robot may need to take a detour to reach a certain place, or it may move through a muddy area causing wheel splippage, etc. Finally, in rural regions communication infrastructure is often lacking or limited and therefore robots can not assume the availability of broadband communication channels.

With these motivations in mind, in this paper we present an algorithm to solve this estimation problem with a team of robots. Coordination between the agents is obtained with minimal information exchange and by leveraging the mathematical properties of GPs to promote dispersion. The approach is fully distributed and each robot just broadcasts to the rest of the team the limited information consisting of the locations where it has collected data, the value it measured, and its unique identifier—no more than a handful of bytes at a very low frequency. No other communication is required and robots never exchange their individual plans or models. In addition, each robot uses a refinement of our recently developed planner for stochastic orienteering to ensure that it reaches the final location before it runs out energy. Through extensive simulations we demonstrate that this approach ensures robots collect samples in areas leading to a more accurate reconstructions of the underlying unknown scalar field.

The rest of the paper is organized as follows. Selected related work is presented in section 2. The problem formulation is introduced in section 3 and our methods are discussed in section 4. In section 5 we present extensive simulations to evaluate our proposal and in section 6 we draw the conclusions.

2 Related Work

The problem considered in this contribution is related to the *orienteering* combinatorial optimization problem where one has to plan a path through a weighted graph to gather the maximum sum of vertex rewards while ensuring that the length of the path does not exceed a preassigned travel budget. Recently, we have extensively studied this problem in agricultural settings, both for single agents [7, 13] and multiple agents [14], and we also considered its stochastic variants [12]. In all these works, however, rewards associated with vertices were static and assigned upfront, while in this work rewards associated with vertices are iteratively re-estimated based on the gathered samples. Moreover, our former multi-robot solution [14] was centralized, while we here propose a fully distributed approach.

The use of Gaussian Processes for estimating scalar fields with robots has also been explored in the past. Suryan and Tokekar [11] proposed an algorithm to reduce in minimal time the variance of the scalar field being estimated with a

GP. Their solution uses a team of robots, but does not consider travel budgets, i.e., robots can travel as much as needed. Similarly, Di Caro et al. [2] propose an estimation algorithm for GPs aiming at reducing uncertainty under time and communication constraints. However, their solution does not consider a travel budget.

Our work is also related to Informative path planning (IPP) where the emphasis is on planning paths to collect the most informative samples [3, 4]. In IPP, however, the set candidate sample points is not given, but is rather determined by the algorithm, and the travel budget is typically not explicitly considered.

3 Problem Statement

We consider the problem of estimating a scalar function $f : \mathcal{X} \to \mathbb{R}$ defined over a bounded region of interest $\mathcal{X}$ given a limited number of observations collected by multiple robots. The goal is to determine where robots should collect these samples. As commonly done in this domain, a graph structure is used to model the navigable environment. We assume that observations of the underlying scalar function can be collected at a limited set of sampling locations, denoted as the set of vertices V in the graph. This assumption holds in a variety of real-world agricultural applications, where specific points of interest (e.g., sentinel trees that serve as early-indicators of ecosystem health) have been pre-identified, or when the robots sense the environment by leveraging a pre-deployed infrastructure (such as soil sensors implanted in the soil). This assumption is not restrictive; when prior sensing locations are not specified, one can choose V arbitrarily, e.g. as a set of equally-spaced points covering the region of interest, as is typical in *naïve* surveying schemes. We consider this navigable environment as a complete graph, with edge set $E = V \times V$. To each edge $e \in E$ we assign a continuous random variable $c(e)$ representing the movement cost (e.g. energy spent) when the robot traverses the edge. We assume that the probability density functions characterizing these random variables are known.

All robots must begin at an assigned start vertex v_s and end at an assigned goal vertex v_g before they run out of energy. Each robot r_i starts with a travel budget B_i. When the robot traverses an edge e, its budget decreases by the random value drawn from the random variable $c(e)$. For simplicity, we assume that all B_is are the same, but this is not a strict requirement. Each time the robot visits a location $v \in V$, using its onboard sensor(s) it collects a sample of the underlying function f, obtaining a noisy observation $y_v = f(v) + \varepsilon$, where $\varepsilon \sim N(0, \sigma_m^2)$ is measurement noise, Gaussian-distributed with zero mean and variance σ_m.

With regard to communication, we make two assumptions: When robots are collecting data, they can only anonymously broadcast packets of the type (v, y_v, n) indicating that they collected observation y_v at vertex v. The last component n is a unique id assigned to each robot— its use will be presented in section 4. At the end of the mission, after the robots have converged at the goal vertex v_g and are in proximity, they can exchange all the data they have gathered

during the mission. However, at that point data collection is concluded and they cannot return to the field and acquire more data to improve the estimate. These communication assumptions are consistent with contemporary technology used by robots in agricultural domain. In particular, LoRa [10] offers the capability of streaming limited amounts of data at long distances and is compatible with the assumptions we made. When robots terminate their mission and are in close proximity data can be instead be exchanged using onboard WiFi.

To reconstruct the scalar field we use Gaussian process (GP) regression, as detailed in section 4. Throughout the mission, using the available data (either collected or communicated) robots can make predictions about the value of f at arbitrary locations in $\mathcal{X}$. We indicate such predictions as $\hat{f}$. The overall objective is to collect the set of observations providing the most accurate reconstruction of the underlying scalar field. As common in estimation literature, in this work our metric for accuracy is the mean squared error (MSE) defined as

$$MSE = \frac{1}{|\mathcal{X}|} \int_{\mathcal{X}} (f(\psi) - \hat{f}(\psi))^2 d\psi.$$

4 Methods

4.1 Gaussian Process Regression

We provide the necessary notation and background for GP regression and we refer the reader to [6] for a more comprehensive introduction. As per our problem definition, we describe the spatial distribution of an unknown parameter (moisture, nitrates, etc.) as a function: $f \colon \mathcal{X} \to \mathbb{R}$ that is continuous in the 2-D environment $\mathcal{X} \subset \mathbb{R}^2$ where measurements are taken. The function which describes the environmental field f and measurement noise σ_m are represented as unique random variables that follow an i.i.d. Gaussian distribution with zero mean μ and variance σ^2. The Gaussian process assumption is to model f as a random probability distribution over a set of functions, and that the value of f for arbitrary inputs x and x' ($f(x)$ and $f(x')$, respectively) has a jointly-Gaussian distribution. We assume that $f(\mathbf{x})$ is a realization of a Gaussian process, which is completely defined by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$ with input vector $\mathbf{x}$:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{1}$$

The joint distribution of observations (the explanatory variable) $\mathbf{y}$, $\{f(x_1) + \varepsilon_1, \ldots, f(x_n) + \varepsilon_n\}$ at inputs $\mathbf{X}$, $\{x_1, \ldots, x_n\}$ and function values (the response variable) $\mathbf{f}$, $\{f_\star, \ldots, f_{\star^n}\}$ can be written as:

$$\begin{bmatrix} \mathbf{y} \\ f(x_\star) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}_N & k\left(\mathbf{X}, x_\star\right) \\ k\left(x_\star, \mathbf{X}\right) & k\left(x_\star, x_\star\right) \end{bmatrix}\right) \tag{2}$$

where $\mathbf{y}$ is a column vector of scalar outputs y, from a training set $\mathcal{D}$ of n observations, $\mathcal{D} = (X, \mathbf{y}) = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$. k is the covariance function

(or kernel), σ_n^2 is the variance of the observation noise, and input vectors $\mathbf{x}$ and query points $\mathbf{x}_\star$ of dimension d are aggregated in the $d \times n$ design matrices X and $X_\star$ respectively.

Through the marginalization of jointly Gaussian distributions, we can derive the following predictive conditional distribution at a single query point $f_\star \mid \mathcal{D}, x_\star \sim \mathcal{N}\left(\mathbb{E}\left[f_\star\right], \mathbb{V}\left[f_\star\right]\right)$ as [6]:

$$\mu = \mathbb{E}\left[f_\star\right] = k\left(x_\star, \mathbf{X}\right)\left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}_n\right]^{-1} \mathbf{y} \tag{3}$$

$$\sigma = \mathbb{V}\left[f_\star\right] = k\left(x_\star, x_\star\right) - k\left(x_\star, \mathbf{X}\right)\left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}_n\right]^{-1} k\left(\mathbf{X}, \mathbf{x}_*\right) \tag{4}$$

where $k(\mathbf{X}, \mathbf{X})$ is a matrix containing the joint prior distribution of covariances of the function f at inputs $\mathbf{X}$ and $k\left(x_\star, \mathbf{X}\right)$ is a matrix containing the covariances between the function at query points and training inputs. The covariance function k (or kernel), captures prior knowledge about the function of interest, including stationarity and smoothness. Often, it is assumed that the covariance of any two random variables depends only on their distance (isotropy), independent of their location (stationarity) [6]. The variance in Equation (4) can be computed for any point and not only for the observed locations. This allows to reason about overall map uncertainty at unobserved locations and is key for the robots to decide where to sample next to decrease the MSE.

4.2 Spatial prior

The kernel k establishes a prior likelihood over the space of functions that can fit observed data in the regression task. Kernel selection and tuning is a key component in GP regression tasks. In machine learning the radial basis function (RBF) kernel is often used. However, in this paper, we use the Matérn kernel with $\nu = 3/2$ which is a finitely-differentiable function. Our choice of this kernel is motivated by its broad use in the geostatistical literature for modeling physical processes [9] like those motivating this research. The Matérn covariance function takes the form:

$$K_{\text{Matern}}(X, X_\star) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{l} r\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{l} r\right) \tag{5}$$

where K_ν is a modified Bessel function , $\Gamma(\cdot)$ is the Gamma function, and r is the Euclidean distance between input points X and $X_\star$. The hyperparemeters $\nu > 0$, $l > 0$, and $\sigma^2 > 0$ represent smoothness, lengthscale, and observation variance respectively. As common in GP inference, to account for the measurement noise, the kernel we use is the sum of two kernels, namely the Matérn kernel and a noise term, i.e., the kernel we use is

$$K(X, X_\star) = K_{\text{Matern}}(X, X_\star) + \sigma_n^2 \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix and the term σ_n^2 models the measurement noise σ_m^2. While we keep ν fixed at 3/2, the other hyperparameters $\theta = \{\sigma^2, \sigma_n^2, l\}$ can be trained using various optimization methods using the marginal likelyhood to match the properties of environment and the sampled data [6]. In particular, the length scale l is related to the lag parameter of the *variogram*, a function used in geostatistics that establishes how quickly the variance increases as a function of separation distance between pairs of observations in space [9]. In the GP kernel, smaller values of l imply that variance quickly grows with distance, while with larger values the variance grows less. As we will see in the next subsection, by putting constraints on the range of possible values of l one can implicitly encourage dispersion between the robots, thus promoting the collection of samples in different areas of the environment.

4.3 Exploration

In this section we present the planning algorithm executed by each robot in the team. No global data structure is shared among the agents, and all the quantities described in the following are local to each robot. Let $G = (V, E)$ be the graph of possible sampling locations and let $D = (v_i, y_i)\ i = 1 \dots n$ the set of collected samples (vertices and values). All robots start from the same start vertex v_s and must end at the same goal vertex v_g. D is initialized as an empty set, but then grows as the robot collects more data or receives data from other agents. Each robot is given a unique numerical identifier n_i, but the robots need not to know how many agents are in the team. At each iteration the robot assigns a reward to each of the vertices in V, i.e., it computes a function $r : V \to \mathbb{R}$ assigning a value to each possible sampling location. Different options for the function r will be discussed in the next subsection.

Once the function r has been computed, the robot is faced with an instance of the stochastic orienteering problem, i.e., it has a graph $G = (V, E)$ with known deterministic rewards r associated to vertices and stochastic costs c associated to edges, as well as a residual budget B. At this point the robot executes the algorithm presented in [12] to solve the stochastic orienteering problem (SOP). Because of the intrinsic computational complexity of the orienteering problem, the SOP algorithm uses a Monte Carlo tree search informed by an heuristic aiming at identifying vertices with high value r, low travel cost, and from which the robot can still reach v_g with high probability (the reader is referred to [12] for all details). The SOP algorithm returns the next vertex v_a to visit. The robot then moves to v_a, collects an observation y_a, updates D, and broadcasts the packet (v_a, y_a, n_i) to all other agents, where n_i is the unique id of the robot. This process continues until the SOP algorithm returns v_g, in which case the agent moves to the goal vertex v_g and terminates. Throughout the process the robot keeps listening for possible packets broadcast by other agents, and when they are received the location and sampled values are added to D.

As the SOP algorithm was developed for the single robot case, in this work we added a minor modification to account for the presence of multiple robots. The change is as follows: When considering the reward of a vertex $r(v)$, the robot

considers for all other agents, the last packet they transmitted (if it exists). Then, if it determines that another agent is closer to v than itself, it discounts the reward $r(v)$. More precisely, let v be the vertex whose utility is being evaluated, and $r(v)$ its reward. Let v_c be the location of the current robot, and assume that it determines that robot j has broadcast a packet indicating it collected a sample at vertex v_j. If v_j is closer to v than v_c, then $r(v)$ is updated as $r'(v) = r(v)d(v_j, v)/d(v_c, v)$ where d is the Euclidean distance between the vertices. The rationale for this update is that if another robot is closer to v, then it is more likely to reach v than the former robot, so the utility of v is decreased for the former robot to prevent having both robots visiting v, as this would be a replicated effort wasting resources. However, the utility is not set to zero because robots do not communicate with each other and do not know their individual intentions. Also, since each robot maintains its own set of GP hyperparameters (see discussion below) and these will be different from each other, robots cannot make absolute predictions about the intentions of other robots in the team.

Remark: one could imagine that after a robot has determined which vertex v it will visit next, it could broadcast this information to other agents so that they do not consider it anymore among their choices. However, this is not done for two reasons. Fist, such additional communication would almost double the amount of transmitted data, thus going against our effort to keep exchanged information at a minimum. Second, because of the stochastic nature of the environment there is no guarantee that a robot electing to visit a certain vertex will eventually reach it and collect a sample. Hence we opt for the current approach where robots share measured data only after they have reached and sampled a location.

4.4 Vertex quality computation

Key to the presented approach is the reward function $r : V \to \mathbb{R}$ used by the SOP algorithm to decide which vertex to visit next. Ideally, the function should identify *instrumentally good* vertices to visit, where good in this case means vertices that will yield a reduction of the MSE metric. Different metrics have been proposed in literature. One obvious choice is to use Eq. (4) to predict the variance of vertices in V and set $r(v) = \sigma^2(v)$. In this case, the objective is to assign high values to vertices with high uncertainty in the estimate. In [8] the authors instead propose to use a linear combination of the mean and standard deviation predicted by Eqs. (3) and (4). Their approach aims at discovering the extrema of an unknown function. As in our application we are interested in the entire function, and not just its peaks, we could set $r(v) = |\mu(v)| + \beta\sigma(v)$. Finally, in [1] the authors propose an algorithm to compute the mutual information for vertices (prior to and after being added to the movement graph) using predictions for mean and variance. After having implemented these three alternatives, preliminary experiments did not outline significant differences between them. However, setting $r(v) = \sigma^2(v)$ has the advantage of not requiring the tuning of additional parameters, as it is instead necessary for the other two methods. Therefore, informed by these preliminary findings, in our implementation each robot assigns the predicted variance as the value of a vertex. Note that for vertices already in

D the algorithm sets $r(v) = 0$, so that robots never consider again vertices that have been already sampled at least once.

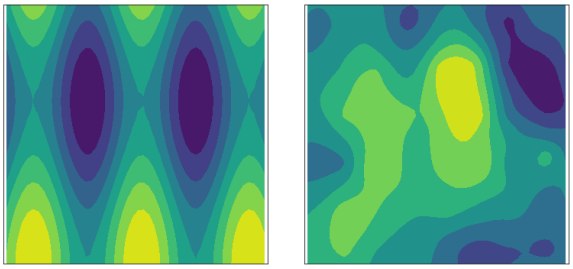
The kernel we use to make predictions about the variance depends on three hyperparameters $\theta = \{\sigma^2, \sigma_n^2, l\}$ that can be tuned to best fit the data in D. As pointed out in [6] Ch.5, to obtain better results it is possible to repeat the optimization process multiple times, with random restarts to avoid getting stuck in suboptimal local minima. In our approach, before assigning values to the vertices each robot executes the optimization locally with ten restarts, but never communicates the hyperparameters of its internal model θ to the other team members. Each agent then operates with its separate set of hyperparameters θ that are unlikely to match the others, due to the random restarts of the optimizer. This difference will further decrease the likelyhood that multiple agents will select the same vertices to sample, because even with identical sets $\mathcal{D}$ the variance predicted by the GP will be different. However, during the optimization process each agent uses the same lower bound l_0 for the length scale l. This choice encourages robots to disperse because the variance of vertices in V near to vertices already inserted in D is lower than the variance of vertices far from D and thereby the reward associated to vertices near to already sampled locations is lower.

5 Experimental Evaluation and Discussion

To establish merits and limitations of the proposed approach, we perform simulations on test cases while varying the different parameters related to the planning and surveying objectives. Due to the limitation of space, we examine the task of reconstructing two scalar fields. The first is a synthetic scalar field with a periodic trend depicted in Figure 1a. The second, displayed in Figure 1b, shows the soil moisture distribution measured in Summer 2018 in a commercial vineyard located Central California. This second scalar field was used as benchmark in some of our former publications [14]. To ease the comparisons between the two cases, both fields were rescaled to the same size, although the amplitude of the underlying values are different. GP predictions of each respective field were made with a Matérn kernel with $\nu = 3/2$. It should be noted that this kernel is commonly used in geostatistical applications and is more appropriate for the soil moisture dataset. Here, the periodic synthetic field serves as a pathological example, with a mismatched spatial prior. In fact, the use of periodic kernels could lead to better results for the synthetic field. Future work will examine online adaptive kernel selection through Bayesian optimization.

Our algorithm, indicated as *Coord* in the following discussion, is compared with two baseline alternatives:

- The random waypoint selection algorithm (RWP), which selects the next vertex to visit at random among those still to be visited. Due to the nature of the selection process, the ability to communicate during the sampling process is immaterial. The RWP algorithm is often considered as a baseline comparison in this type of tasks (see e.g., [2]).



(a) Synthetic dataset (b) Soil moisture dataset

Fig. 1: Benchmark scalar fields to be estimated. With reference to the travel budget B, the length of the side edge is 5. In both instances the start vertex v_s is in the lower left corner and the goal vertex v_g is in the top right corner.

– A non-coordinated (NC) approach, which selects the next sampling point using the same criteria used by our proposed algorithm, but does not exchange any information during the sample process, i.e., during the selection process each agent only considers the samples it collected, but not those collected by the other agents.

At the end of the mission, when all robots have reached v_g, both RWP and NC share all collected sensor observations and the MSE is computed after fitting the GP using all data collected by all robots. This step is not necessary for Coord because data is exchanged on the go, but it ensures that the MSE evaluation is done fairly among the three algorithms. After the algorithm has selected the next point to visit, all algorithms, including Coord, use the same planning engine, i.e., the one we proposed in [12]. Finally, both NC and Coord do refit of the GP and update the parameters θ before computing r, while RWP does not do this step because it does not use the current estimate to select the next point to visit.

All procedures were executed single-threaded in Python running on an Apple M1 processor. All computations related to GP fitting and processing use the scikit-learn library [5]. For both scalar fields considered in the tests, we varied the number of agents (3,5,7,9), the budget B (10,15,20), and the parameter l_0 (0.1,0.5,1). For each combination of parameters, twenty independent simulations were performed, for a total of about 3500 executions.

Figure 4 show the average MSE as a function of the number robots for all algorithms. As expected, the trend is decreasing (i.e. improved prediction accuracy) with diminishing returns as the number of robot grows. We can observe that the proposed algorithm outperforms the others. Note that the range of values for MSE in the two test cases is different because the of the different values in the underlying scalar fields. Next, in Figure 3 we show the reconstructed scalar field for the three algorithms with a budget of 20 and 5 robots. The red dots show the locations where the samples were collected. Due to the random selection process, RWP ends up collecting less samples before exhausting the

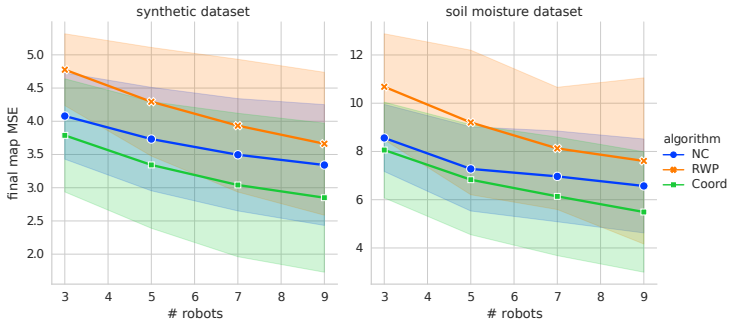


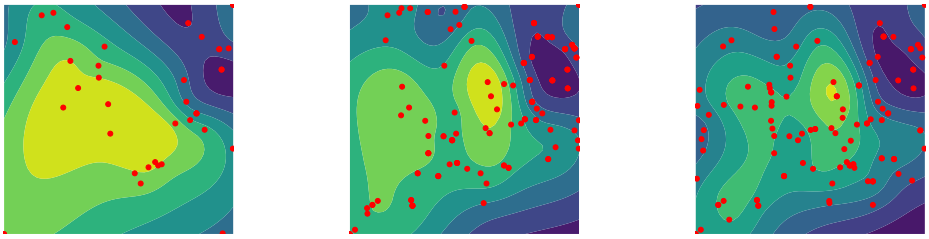
Fig. 2: Average final map MSE for n=20 trials per algorithm. Error bars show $\pm$ one standard deviation.

budget and this leads to an inferior estimate. NC and Coord, instead, collect more samples, but we can see how Coord spreads them more uniformly and ultimately leads to a more accurate estimate (see Figure 1b for the ground truth). Similar results are observed for the synthetic map, but are not displayed for lack of space. Table 1 provides a more detailed numeric analysis of the performance of the three algorithms. Specifically, we look at the number of unvisited locations as well as the number of locations visited by more than one robot. These are two proxies for the MSE metric, and lower values are desired in both cases. When the travel budget is 10, the number of unvisited locations is similar for the three algorithms because with limited budget the set of available choices before the budget is used is limited. As the budget grows, we see that the Coord algorithm emerges as the algorithm with less unvisited vertices, thus substantiating the claim that agents spread in the environment in a more coordinated fashion. For the number of revisited locations, RWP (as expected) always has the lowest number of revisited locations, due to the completely random nature of the selection. However, when comparing NC with Coord we see that the latter has always a lower number, again showing that the agents better spread in the environment avoiding to revisit the same places, thus ensuring that coordination leads to a better use of the robots are mobile sensors.

map	synthetic dataset			soil moisture dataset		
budget	10	15	20	10	15	20
RWP	179/6	164/8	155/10	179/6	164/9	155/10
NC	164/17	139/31	122/42	166/16	140/30	122/40
Coord	163/12	129/18	104/21	166/11	133/17	106/21

Table 1: Average number of unvisited and re-visited waypoints, from an experimental setting of 200 candidate sampling locations. X/Y means that there were on average X unvisited vertices and Y revisited vertices.

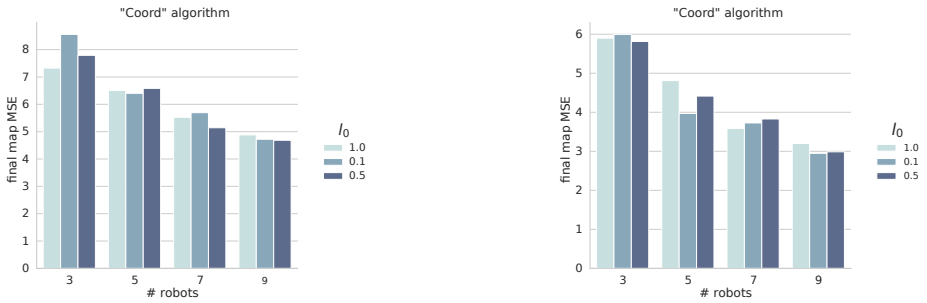
Finally, in Figure 4 we display how the choice of l_0, the lower-bound for the length scale parameter l, impact the value of the MSE metric. The two panels



(a) RWP (b) NC (c) Coord

Fig. 3: Reconstructed scalar field soil moisture map with 5 robots and $B = 20$.

correspond to a budget of 15 and 20 respectively, and group the results for different numbers of surveying robots. For 9 robots the impact is marginal, and this is explained by the fact that with this many agents the team manages to cover most of the environment during the mission. However, for budget of 15 and 3 robots, a value of $l_0 = 1$ gives a clearly better result. Likewise, for budget of 20 and 5 robots, a value of $l_0 = 0.5$ is best. These results show that by tuning l_0 it is possible to implicitly promote better dispersion in the team and then lower values for the MSE. An outstanding question to be investigated is how to select this value in a general setting. Nevertheless, these results confirm our hypothesis that by constraining the GP kernel parameters being optimized, one can enforce different behaviors on the team members.



(a) Budget=15 (b) Budget=20

Fig. 4: Average MSE for different values of l_0 and number of robots.

6 Conclusions

We have presented an approach to reconstruct a scalar field using a team of robots performing distributed GP estimation. Through limited communication,

by exploiting the underlying properties of GPs, robots implicitly manage to disperse thorough the domain and collect samples at locations leading to a more accurate estimation. Our proposed approach has been extensively evaluated in simulation and outperforms competing approaches.

References

1. E. Contal, V. Perchet, and N. Vayatis. Gaussian process optimization with mutual information. In *Proceedings of the 31st International Conference on Machine Learning*, pages 253–261, 2014.
2. G.A. Di Caro and Abdul W. Ziaullah Y. Map Learning via Adaptive Region-Based Sampling in Multi-robot Systems. In F. Matsuno, S. Azuma, and M. Yamamoto, editors, *Distributed Autonomous Robotic Systems*, pages 335–348. Springer, 2022.
3. Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, August 2014.
4. M.G. Jadidi, J.V. Miro, and G. Dissanayake. Sampling-based incremental information gathering with applications to robotic exploration and environmental monitoring:. *The International Journal of Robotics Research*, 38(6):658–685, April 2019.
5. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
6. C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 2006.
7. A. Shamshirgaran and S. Carpin. Reconstructing a spatial field with an autonomous robot under a budget constraint. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022 (to appear).
8. N. Srinivas, A. Krause, S.M. Kakade, and M.W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.
9. M.L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer-Verlag, New York, 1999.
10. J. P. S. Sundaram, W. Du, and Z. Zhao. A survey on LoRa networking: Research problems, current solutions, and open issues. *IEEE Communications Surveys & Tutorials*, 22(1):371–388, 2019.
11. V. Suryan and P. Tokekar. Learning a spatial field in minimum time with a team of robots. *IEEE Transactions on Robotics*, 36(5):1562–1576, 2020.
12. T.C. Thayer and S. Carpin. Solving stochastic orienteering problems with chance constraints using monte carlo tree search. In *Proceedings of the IEEE International Conference on Automation Science and Engineering*, 2022 (to appear).
13. T.C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin. Routing algorithms for robot assisted precision irrigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2221–2228, 2018.
14. T.C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin. Multi-robot routing algorithms for robots operating in vineyards. *IEEE Transactions on Automation Science and Engineering*, 17(3):1184–1194, 2020.