EFFICIENT BAYESIAN SENSOR PLACEMENT AND INFORMATIVE PATH PLANNING

by

Kalvik Jakkala

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing & Information Systems

Charlotte

2024

Approved by:

_____
Dr. Srinivas Akella

_____
Dr. Jaya Bishwal

_____
Dr. Minwoo Lee

_____
Dr. Wenhao Luo

_____
Dr. Gabriel Terejanu

©2024
Kalvik Jakkala
ALL RIGHTS RESERVED

ABSTRACT

KALVIK JAKKALA. Efficient Bayesian Sensor Placement
and Informative Path Planning. (Under the direction of DR. SRINIVAS AKELLA)

Sensor placement and Informative Path Planning (IPP) are fundamental problems that frequently arise in various domains. The sensor placement problem necessitates finding optimal sensing locations in an environment, enabling accurate estimation of the overall environmental state without explicitly monitoring the entire space. Sensor placement is particularly relevant for problems such as estimating ozone concentrations and conducting sparse-view computed tomography scanning. IPP is a closely related problem that seeks to identify the most informative locations along with a path that visits them while considering path constraints such as distance bounds and environmental boundaries. This proves useful in monitoring phenomena like ocean salinity and soil moisture in agricultural lands—situations where deploying static sensors is infeasible or the underlying dynamics of the environment are prone to change and require adaptively updating the sensing locations.

This thesis provides new insights leveraging Bayesian learning along with continuous and discrete optimization, which allow us to reduce the computation time and tackle novel variants of the considered problems. The thesis initially addresses sensor placement in both discrete and continuous environments using sparse Gaussian processes (SGP). Subsequently, the SGP-based sensor placement approach is generalized to address the IPP problem. The method demonstrates efficient scalability to large multi-robot IPP problems, accommodates non-point FoV sensors, and models differentiable path constraints such as distance budgets and boundary limits. Then the IPP approach is further generalized to handle online and decentralized heterogeneous multi-robot IPP. Next, the thesis delves into IPP within graph domains to address the methane gas leak rate estimation and source localization problem. An efficient

Bayesian approach for leak rate estimation is introduced, enabling a fast discrete optimization-based IPP approach. Lastly, the thesis explores sensor placement in graph domains for wastewater-based epidemiology. A novel graph Bayesian approach is introduced, facilitating the placement of sensors in wastewater networks to maximize pathogen source localization accuracy and enable efficient source localization of pathogens.

ACKNOWLEDGEMENTS

This work was funded in part by the UNC Charlotte Office of Research and Economic Development, the Graduate School at the University of North Carolina at Charlotte, and by NSF under Award Number IIP-1919233.

TABLE OF CONTENTS

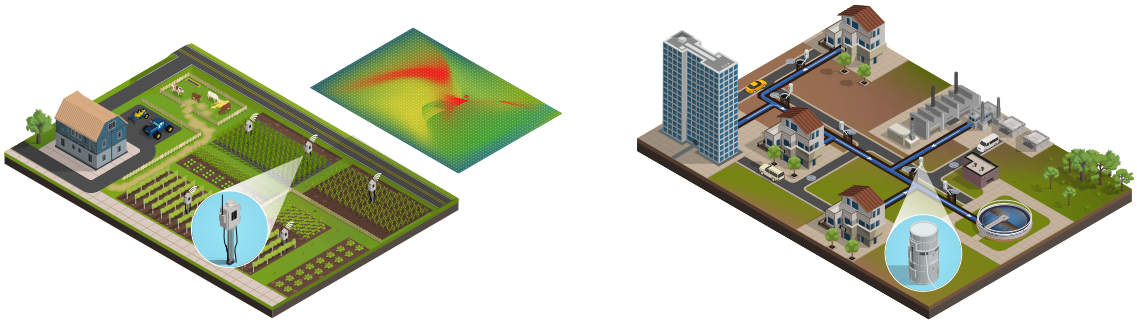
LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

This thesis addresses two closely related problems: sensor placement and informative path planning. Each problem is further partitioned into data field estimation and source localization problems. Bayesian learning, along with discrete and continuous optimization methods, are leveraged to develop computationally fast approaches to address these problems.

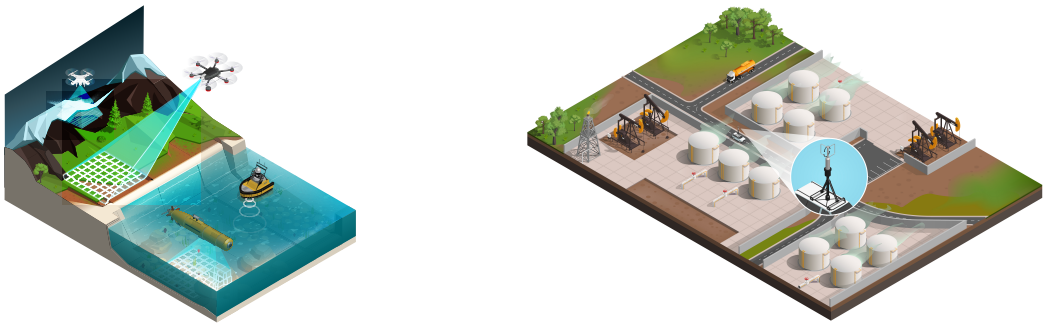
Sensor placement: Numerous real-world tasks, including environmental monitoring, infrastructure monitoring, and agriculture, necessitate the observation of phenomena like ozone concentration, ocean water salinity, soil moisture, and temperature. However, the cost and feasibility constraints often render it impractical to monitor the entire environment with a dense sensor network. Hence, our objective is to identify strategic locations for a limited set of sensors, ensuring that the data they provide yields the most accurate estimate of the phenomenon across the entire environment. This problem is commonly referred to as the sensor placement problem. Figure 1.1 illustrates sensor placement for environment monitoring and source localization.



(a) (b)

Figure 1.1: (a) Illustration of sensor placement for environment monitoring. (b) Illustration of sensor placement in a wastewater network to localize virus outbreaks.

Informative path planning: Finding the most informative path for data col-



(a) (b)

Figure 1.2: (a) Illustration of informative path planning for environment monitoring. (b) Illustration of informative path planning for gas leak source localization.

lection is known as the Informative Path Planning (IPP) problem. The problem is, at its core, a sensor placement problem with path constraints, i.e., finding the most informative sensing locations while also ensuring that we satisfy any path constraints such as a distance budget and boundary limits. Figure 1.2 illustrates informative path planning for environment monitoring and source localization.

Moreover, both of the problems mentioned above have multiple variants depending on factors such as the type of environment—discrete, continuous, or graph—and the type of sensors—point, non-point, or integrated. Here, point sensors have a point field-of-view (FoV), such as a temperature probe; non-point sensors have a non-point FoV, such as a thermal vision camera, and integrated sensors integrate the data within their FoV, as seen in a tunable diode laser spectrometer (TDLAS) gas sensor. Therefore, one must explicitly consider such factors while developing solution approaches to address these problems.

1.1 Thesis Contributions

1. **Sensor Placement in Continuous and Discrete Environments for Data Field Estimation**

   (a) We present an approach to obtain sensor placements for monitoring spatially (or spatiotemporally) correlated phenomena in continuous spaces.

Our approach formulates the sensor placement problem as a regression problem using SGPs.

(b) Our approach can even be used for sensor placement in discrete environments. We present an efficient assignment problem-based method to map our continuous space solutions to discrete solution spaces.

2. **Informative Path Planning in Continuous and Discrete Environments for Data Field Estimation**

   (a) We present an efficient informative path planning (IPP) approach using sparse Gaussian processes for spatially and spatio-temporally correlated environments. Our approach does not discretize the environment; it instead optimizes paths in a continuous space using efficient gradient-based approaches.

   (b) We present an approach to model routing constraints such as a distance budget and the environment's boundary limits.

   (c) Our approach can accommodate both discrete and continuous sensing robots, with both point and non-point sensing FoV shapes, and can generate smooth paths.

   (d) Our approach can plan informative paths for multiple robots simultaneously.

3. **Online and Decentralized Heterogeneous Multi-Robot Informative Path Planning in Continuous and Discrete Environments for Data Field Estimation**

   (a) We generalize our offline IPP approach to accommodate online multi-robot IPP by employing streaming sparse Gaussian processes.

(b) Our approach can also model heterogeneous multi-robot IPP, wherein each robot can possess a different sensing model and path constraints.

(c) Our approach employs a K-nearest neighbor-based environment partitioning scheme to effectively enable decentralized multi-robot IPP with minimal communication requirements.

4. **Informative Path Planning in Graphs for Source Localization**

(a) Presents a fast and effective Bayesian approach for leak rate estimation from gas concentration data.

(b) Derives an efficient analytical solution to an information metric—Expected Entropy Reduction (EER)—that is used in the IPP problem.

(c) Improves the runtime efficiency of the Generalized Cost-benefit (GCB) algorithm used to solve the IPP problem.

(d) Introduces an arc routing variant of the GCB algorithm for IPP in graph networks.

5. **Sensor Placement in Graphs for Source Localization**

(a) Introduces an approach to model a reduced graph representation of wastewater networks ideal for efficient sensor placement optimization and source localization.

(b) Presents a novel approach to use the inherent structure of wastewater networks to build graphical Bayesian networks.

(c) Establishes a Bayesian optimization objective that can be used to efficiently find ideal sensor placements for accurate virus source localization. The method also ensures that the placements collect high-concentration wastewater samples.

(d) Presents a computationally efficient Bayesian source localization approach.

(e) Empirically establishes that our optimization objective is submodular, giving us a $(1 - 1/e)$ approximation factor on our solution placements.

1.2 Preliminaries

A key component of sensor placement and informative planning approaches is being able to quantify the informativeness of sensing locations. We use Bayesian methods to define distributions over quantities of interest (e.g., the temperature at each location in an environment) and quantify the informativeness of locations using information metrics such as entropy and mutual information. This section details Gaussian distributions, their infinite-dimensional extension—Gaussian processes, sparse Gaussian processes, and mutual information.

1.2.1 Gaussian Distributions

A random variable $y \in \mathbb{R}^d$ is said to have a Gaussian distribution with mean μ and covariance Σ if its probability density function is given by the following:

$$p(y|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{\frac{-1}{2}((x-\mu)^T\Sigma^{-1}(x-\mu))} \tag{1.1}$$

A multivariate Gaussian distribution's mean and covariance matrices can be partitioned. Consider the Gaussian distributed random variables $\mathbf{y}$, with mean μ and covariance matrix $\mathbf{\Sigma}$:

$$\mathbf{y} = \begin{bmatrix} y_A \\ y_B \end{bmatrix} \quad \mu = \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix} \tag{1.2}$$

It has the following key properties:

1. **Normalization:** The integral of a Gaussian distributed random variable is equal to one:

$$\int_{\mathbf{y}} p(\mathbf{y}) d\mathbf{y} = 1 \tag{1.3}$$

2. **Summation:** The distribution of the sum of two random variables follows a Gaussian distribution if both random variables are Gaussian distributed:

$$\begin{aligned} y_A &\sim \mathcal{N}(\mu_A, \Sigma_{AA}) \\ y_B &\sim \mathcal{N}(\mu_B, \Sigma_{BB}) \\ y_A + y_B &\sim \mathcal{N}(\mu_A + \mu_B, \Sigma_A + \Sigma_B) \end{aligned} \tag{1.4}$$

3. **Conditioning:** Conditioning Gaussian distributed random variable results in a Gaussian distributed posterior. Moreover, the posterior's mean and covariance can be calculated analytically; this is not always possible for most other distributions:

$$\begin{aligned} p(y_A|y_B) &= \frac{p(y_A, y_B)}{\int_{y_A} p(y_A, y_B)} = \frac{p(y_B|y_A)p(y_A)}{\int_{y_A} p(y_A, y_B)} \\ y_A|y_B &\sim \mathcal{N}(\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(y_B - \mu_B),\ \Sigma_{AA} - \Sigma_{AB}\Sigma^{-}1_{BB}\Sigma_{BA}) \end{aligned} \tag{1.5}$$

4. **Marginalization:** Marginalization of a variable from a multivariate Gaussian distribution results is a Gaussian distributed posterior:

$$\begin{aligned} p(y_A) &= \int_{y_B} p(y_A, y_B) dy_B \\ y_A &= \mathcal{N}(\mu_A, \Sigma_{AA}) \end{aligned} \tag{1.6}$$

1.2.2 Gaussian processes (GPs)

Gaussian processes [2] are a non-parametric Bayesian approach that we can use for regression, classification, and generative problems. GPs can also be viewed as infinite dimensional extensions of Gaussian distributions, in which, instead of considering a

distribution over a finite number of random variables, we consider an infinite number of random variables, giving us a distribution over functions.

Suppose we are given a regression task's training set $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, ..., n\}$ with n data samples consisting of inputs $\mathbf{x}_i \in \mathbb{R}^d$ and noisy labels $y_i \in \mathbb{R}$, such that, $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. Here σ^2_{noise} is the variance of the independent additive Gaussian noise in the observed labels y_i, and the latent function $f(\mathbf{x})$ models the noise-free function of interest that characterizes the regression dataset.

GPs model such datasets by assuming a GP prior over the space of functions that we could use to model the dataset, i.e., they assume the prior distribution over the function of interest $p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(0, \mathbf{K})$, where $\mathbf{f} = [f_1, f_2, ..., f_n]^\top$ is a vector of latent function values, $f_i = f(\mathbf{x}_i)$. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^\top$ is a vector (or matrix) of inputs, and $\mathbf{K} \in \mathbb{R}^{n \times n}$ is a covariance matrix, whose entries $\mathbf{K}_{ij}$ are given by the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$.

The kernel function parameters are tuned using Type II maximum likelihood [3] so that the GP accurately predicts the training dataset labels. We can compute the posterior of the GP with the mean and covariance functions:

$$\begin{aligned} m_{\mathbf{y}}(\mathbf{x}) &= \mathbf{K}_{xn}(\mathbf{K}_{nn} + \sigma^2_{\text{noise}} I)^{-1}\mathbf{y}\,, \\ k_{\mathbf{y}}(\mathbf{x}, \mathbf{x}') &= k(\mathbf{x}, \mathbf{x}') - \mathbf{K}_{xn}(\mathbf{K}_{nn} + \sigma^2_{\text{noise}} I)^{-1}\mathbf{K}_{nx'}\,, \end{aligned} \tag{1.7}$$

where $\mathbf{y}$ is a vector of all the outputs, and the covariance matrix subscripts indicate the variables used to compute it, i.e., $\mathbf{K}_{nn}$ is the covariance of the training inputs $\mathbf{X}$, and $\mathbf{K}_{xn}$ is the covariance between the test input $\mathbf{x}$ and the training inputs $\mathbf{X}$.

Note that even though the method is called Gaussian processes, the Gaussian assumption applies only to the likelihood $p(\mathbf{y}|\mathbf{x})$ and prior distribution $p(\mathbf{x})$. Moreover, there are generalizations of Gaussian processes that leverage approximate inference techniques [4, 5, 6] to accommodate non-Gaussian likelihoods. As such, Gaussian processes are capable of modeling a large class of functions. However, the approach

requires an inversion of a matrix of size $n \times n$, which is a $\mathcal{O}(n^3)$ operation, where n is the number of training set samples. Thus this method can handle at most a few thousand training samples.

1.2.3 Sparse Gaussian processes (SGPs)

Sparse Gaussian processes [7, 8, 9, 10, 5] address the computational cost issues of Gaussian processes. SGPs do this by approximating the full GP using another Gaussian process supported with m data points called *inducing points*, where $m \ll n$. Since the SGP support set (i.e., the data samples used to estimate the training set labels) is smaller than the full GP's support set (the whole training dataset), SGPs reduce the matrix inversion cost to $\mathcal{O}(m^3)$.

There are multiple SGP approaches; one particularly interesting approach is the sparse variational GP (SVGP) [8], a well-known approach in the Bayesian community that has had a significant impact on the sparse Gaussian process literature given its theoretical properties [11, 12].

To approximate the full GP, the SVGP approach uses a variational distribution q parametrized with m inducing points. The approach treats the inducing points as variational parameters instead of model parameters, i.e., the inducing points parametrize a *distribution* over the latent space of the SGP instead of directly parameterizing the latent space. Thus the inducing points are protected from overfitting. The SVGP approach's mean predictions and covariances for new data samples are computed using the following equations:

$$\begin{aligned} m_{\mathbf{y}}^{q}(\mathbf{x}) &= \mathbf{K}_{xm}\mathbf{K}_{mm}^{-1}\boldsymbol{\mu}\,, \\ k_{\mathbf{y}}^{q}(\mathbf{x},\mathbf{x}') &= k(\mathbf{x},\mathbf{x}') - \mathbf{K}_{xm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mx'} + \\ &\qquad \mathbf{K}_{xm}\mathbf{K}_{mm}^{-1}\mathbf{A}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mx'}\,, \end{aligned} \tag{1.8}$$

where the covariance term subscripts indicate the input variables used to compute the covariance; m corresponds to the inducing points $\mathbf{X}_m$ and x corresponds

to any other data point $\mathbf{x}$. $\boldsymbol{\mu}$ and $\mathbf{A}$ are the mean and covariance of the optimal variational distribution q^*. The approach maximizes the following evidence lower bound (ELBO) $\mathcal{F}$ to optimize the parameters of the variational distribution:

$$\mathcal{F} = \underbrace{\frac{n}{2}\log(2\pi)}_{\text{constant}} + \underbrace{\frac{1}{2}\mathbf{y}^\top(\mathbf{Q}_{nn} + \sigma^2_{\text{noise}}I)^{-1}\mathbf{y}}_{\text{data fit}} + \underbrace{\frac{1}{2}\log|\mathbf{Q}_{nn} + \sigma^2_{\text{noise}}I|}_{\text{complexity term}} - \underbrace{\frac{1}{2\sigma^2_{\text{noise}}}Tr(\mathbf{K}_{nn} - \mathbf{Q}_{nn})}_{\text{trace term}}, \tag{1.9}$$

where $\mathbf{Q}_{nn} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}$ and $\mathbf{K}_{mm}$ is the covariance matrix of the inducing points $\mathbf{X}_m$. The lower bound $\mathcal{F}$ has three key terms. The data fit term ensures that the training set labels are accurately predicted. The complexity and trace terms are independent of the labels. The complexity term ensures that the inducing points are spread apart to ensure good coverage of the whole training set, and the trace term represents the sum of the variance of the conditional $p(\mathbf{f}|\mathbf{f}_m)$. Here $\mathbf{f}_m$ are the latent variables corresponding to the inducing point inputs $\mathbf{X}_m$. When the trace term becomes zero, the m solution inducing points become a sufficient statistic for the n training samples, i.e., an SGP with only the m solution inducing points can make the same predictions as a GP with all the n samples in its training set. Please refer to Bauer et al. [11] for an in-depth analysis of the SVGP's lower bound.

1.2.4 Mutual Information

Mutual information (MI) is an information metric that can be used to determine the amount of information overlap between two random variables. MI is derived from the KL divergence, which can be used to quantify the similarity of two distributions, i.e., it is zero if the two distributions are the same. However, it is not symmetric, i.e., $\text{KL}(p||q) \neq \text{KL}(q||p)$, when $p(x)$ and $q(x)$ are different. The KL divergence is defined as follows:

$$\begin{aligned} \mathrm{KL}(p||q) &= -\int p(x)\ln q(x)dx - \left(-\int p(x)\ln p(x)\right) \\ &= -\int p(x)\ln\left\{\frac{q(x)}{p(x)}\right\}dx \end{aligned} \tag{1.10}$$

The MI between the random variables x and y is defined as follows:

$$\begin{aligned} \mathrm{MI}[x,y] &= \mathrm{KL}(p(x,y)||p(x)p(y)) \\ &= -\int\int p(x,y)\ln\left\{\frac{p(x)p(y)}{p(x,y)}\right\}dxdy \\ &= H(x) - H(x|y) \end{aligned} \tag{1.11}$$

Here, H is the entropy function.

Although MI is theoretically well-grounded and ideal for determining sensing locations in sensor placement and informative path planning problems, it is often difficult to compute. This is because of the integrals needed to compute MI can become intractable depending on the distributions over the random variables.

As such, a common approach is to fit a Gaussian process over the random variables and then compute MI between the variables [13]. Although, the approach is still computationally expensive, it is tractable.

CHAPTER 2: Sensor Placement in Continuous and Discrete Environments for Data Field Estimation

2.1 Introduction

Meteorology and climate change are concerned with monitoring correlated environmental phenomena such as temperature, ozone concentration, soil chemistry, ocean salinity, and fugitive gas density [13, 14, 15, 16, 17]. However, it is often too expensive and, in some cases, even infeasible to monitor the entire environment with a dense sensor network. We therefore aim to determine strategic locations for a sparse set of sensors so that the data from these sensors gives us the most accurate estimate of the phenomenon over the entire environment. We address this *sensor placement problem* for correlated environment monitoring.

Moreover, we focus on the sparsely labeled sensor placement problem, wherein only a few labeled data samples are available. This data restriction limits the applicability of many parametric approaches [18], such as deep learning and deep reinforcement learning. The sparsely labeled sensor placement problem is a fundamental problem with diverse and important applications. For example, informative path planning (IPP) is a crucial problem in robotics that involves identifying informative sensing locations for robots while considering travel distance constraints [14]. Similar sensor placement problems arise in autonomous robot inspection and monitoring of 3D surfaces [19], for example, when a robot must monitor stress fractures on an aircraft body. Recently a sensor placement approach has even been used to learn dynamical systems in a sample-efficient manner [20]. These problems require fast solutions, but the sensor placement problem often becomes a computationally expensive bottleneck in current approaches.

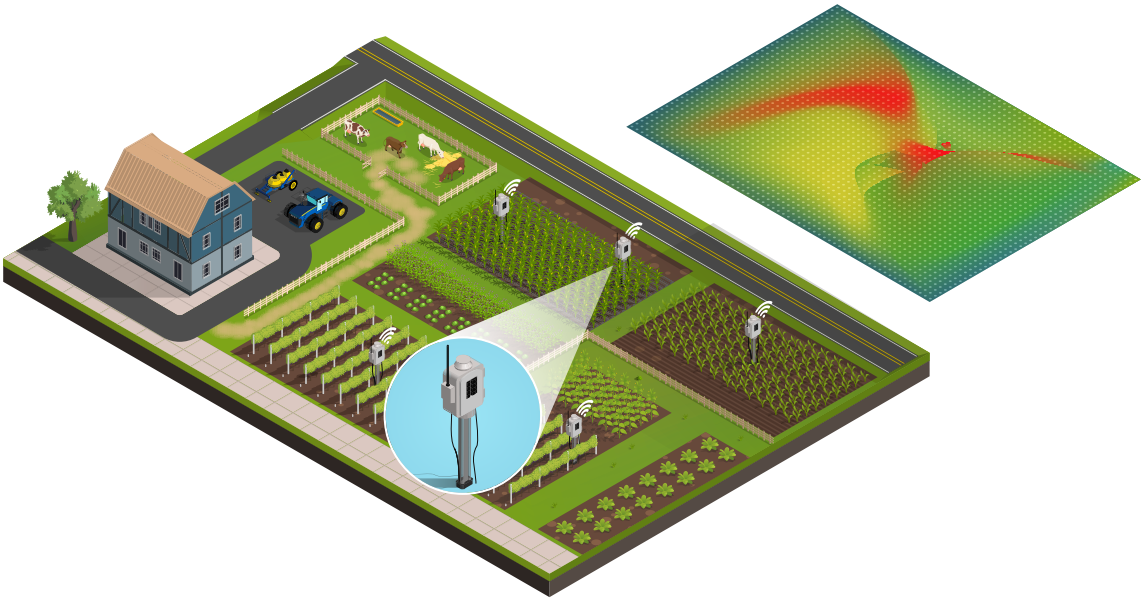


Figure 2.1: Illustration of sensor placement in a farm.

An effective approach to address the considered sensor placement problem is to use Gaussian processes (GPs) [21, 22, 23, 13]. We can capture the correlations of the environment using the GP's kernel function and then leverage the GP to estimate information metrics such as mutual information (MI). Such metrics can be used to quantify the amount of new information that can be obtained from each candidate sensor location. However, computing MI using GPs is very expensive as it requires the inversion of large covariance matrices whose size increases with the environment's discretization resolution. Consequently, these methods do not scale well to continuous spaces and 3D spaces, and have limited applicability when addressing the aforementioned applications, which often necessitate a large number of sensor placements or a fine sensor placement precision that is infeasible with discrete approaches.

Sparse Gaussian processes (SGPs) [24] are a computationally efficient variant of GPs. Therefore, one might consider using SGPs instead of GPs in GP-based sensor placement approaches. However, a naive replacement of GPs with SGPs is not always possible or efficient. This is because SGPs must be retrained for each evaluation of

MI. In sensor placement approaches, MI is often evaluated repeatedly, making SGPs computationally more expensive than GPs for sensor placement. So even though SGPs have been studied for over two decades, SGPs have received limited attention for addressing the sensor placement problem [25].

The objective of this paper is to develop an efficient approach for addressing the sparsely labeled sensor placement problem in correlated environments. We present an efficient, sparsely supervised, gradient-based approach for sensor placement in continuous environments by leveraging the connection between SGPs and sensor placement problems, and the inherent structure of the SGP's optimization function. Unlike most prior approaches, our method is fully differentiable. As such, it can even be incorporated into deep neural networks optimized for other downstream tasks. We also generalize our method to efficiently handle sensor placement in discrete environments. Our approach enables efficient sensor placement in 3D spaces, spatiotemporally correlated spaces, and derivative problems such as informative path planning critical for robotics applications.

2.2 Problem Statement

Consider a correlated stochastic process Ψ over an environment $\mathcal{V} \subseteq \mathbb{R}^d$ modeling a phenomenon such as temperature. The *sensor placement problem* is to select a set $\mathcal{A}$ of s sensor locations $\{\mathbf{x}_i \in \mathcal{V}, i = 1, ..., s\}$ so that the data $y_i \in \mathbb{R}$ collected at these locations gives us the most accurate estimate of the phenomenon at every location in the environment. We consider estimates with the lowest root-mean-square error (RMSE) to be the most accurate. An ideal solution to this sensor placement problem should have the following key properties:

1. The approach should be computationally efficient and produce solutions with low RMSE. Since the environment is correlated, this should also result in the solution sensor placements being well separated to ensure that the sensors collect only novel data that is crucial for accurately reconstructing the data field.

2. The approach should handle both densely and sparsely labeled environments. In a densely labeled environment, we have labeled data at every location in the environment. In a sparsely labeled environment, we have labeled data that is sufficient only to capture the correlations in the environment, or we have domain knowledge about how the environment is correlated.

3. The approach should handle both continuous sensor placements $\mathcal{A} \subseteq \mathcal{V}$, where the sensors can be placed anywhere in the environment, and discrete sensor placements $\mathcal{A} \subseteq \mathcal{S} \subseteq \mathcal{V}$, where the sensors can only be placed at a subset of a pre-defined set of locations $\mathcal{S}$.

2.3 Related Work

Early approaches to the sensor placement problem [26, 27] used geometric models of the sensor's field of view to account for the region covered by each sensor and used computational geometry or integer programming methods to find solutions. Such approaches proved useful for problems such as the art gallery problem [28], which requires one to place cameras so that the entire environment is visible. However, these approaches do not consider the spatial correlations in the environment.

This problem is also studied in robotics [29, 30, 31]. Similar to geometric approaches, authors focus on coverage by leveraging Voronoi decompositions [28]. A few authors [32, 33], have even considered Gaussian kernel functions, but they did not leverage the full potential of Gaussian processes.

Gaussian process (GP) based approaches addressed the limitations of geometric model-based sensor placement approaches by learning the spatial correlations in the environment. The learned GP is then used to quantify the information gained from each sensor placement while accounting for the correlations of the data field. However, these methods require one to discretize the environment and introduce severe computational scaling issues. Our method finds sensor locations in continuous spaces

and overcomes the computational scaling issues.

Early GP-based approaches [22, 23] placed sensors at the highest entropy locations. However, since GPs have high variance in regions of the environment far from the locations of the training samples, such approaches tended to place sensors at the sensing area's borders, resulting in poor coverage of the area of interest. [13] used mutual information (MI) computed with GPs to select sensor locations with the maximal information about all the unsensed locations in the environment. The approach avoided placing the sensors at the environment's boundaries and outperformed all earlier approaches in terms of reconstruction quality and computational cost.

[34] leveraged a full variational Gaussian process to compute mutual information (MI) and then utilized a stochastic greedy algorithm to select the solution sensors that maximize MI. This method is a computationally faster generalization of [13]. However, it still discretizes the environment and performs a combinatorial search, which limits its scalability to problems in 3D spaces.

[16] recently proposed an approach to model spatiotemporal data fields using a combination of sparse Gaussian processes (SGPs) and state space models. They then used the spatiotemporal model to sequentially place sensors in a discretized version of the environment. Although their spatiotemporal model of the environment resulted in superior sensor placements, the combinatorial search becomes prohibitively large and limits the size of the problems that can be solved using their method.

[35] addressed sensor placement in continuous spaces by first mapping the data samples to a lower-dimensional space and then estimating a lower bound for the mutual information among the data. The lower bound on the mutual information was then optimized using Bayesian optimization to find the solution sensing locations. Sensor placement in continuous environments has also been addressed in the context of informative path planning using gradient-free optimization methods such as evolutionary algorithms [36] and Bayesian optimization [37]. However, both of these

approaches maximize MI computed using GPs, which is computationally expensive. In addition, evolutionary algorithms and Bayesian optimization are known to have poor scalability.

A closely related problem is sensor placement with labeled data, where we either have a large corpus of real or simulated data, which is used to select a subset of sensing locations and validate their performance from the reconstruction error on the training set. [38] used random forests, [25] used SGPs and optimized them using genetic algorithms, [39] leveraged reinforcement learning, and [40] used linear programming to address this problem. Although such approaches are capable of finding good sensor placement locations, they do not generalize to novel environments where we do not have any significant amounts of data. Our approach addresses the sparsely labeled sensor placement problem.

2.4 Method

2.4.1 Theoretical Foundation

We first address the SP problem, which requires us to find sensing locations in an environment so that the data collected at these sensing locations will accurately reconstruct the data field being monitored. We treat the underlying data field as a stochastic process that can be modeled using a Gaussian process (GP). Note that GPs are not limited to modeling Gaussian distributed functions; a GP is a set of random variables $\mathbf{f}(\mathbf{X}) = \{f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}\}$ for which any finite subset follows a Gaussian distribution. Here, the latent variables $\mathbf{f}(\mathbf{X})$ represent the noise-free labels of the phenomenon being monitored at every location in the environment.

To find the solution sensor placements, we approximate the abovementioned GP using a sparse distribution q, which introduces m auxiliary inducing latent variables $\mathbf{f}_m(\mathbf{X}_m)$:

$$\begin{aligned} \hat{\mathbf{X}} &= \{\mathbf{X}, \mathbf{X}_m\}, \\ q(\hat{\mathbf{f}}(\hat{\mathbf{X}})) &= p(\mathbf{f}(\mathbf{X}), \mathbf{f}_m(\mathbf{X}_m)) \\ &= p(\mathbf{f}(\mathbf{X})|\mathbf{f}_m(\mathbf{X}_m))q(\mathbf{f}_m(\mathbf{X}_m)). \end{aligned} \tag{2.1}$$

Here, the sparse distribution q is formulated so that all the information relevant to predicting the latent variables $\mathbf{f}$ is captured by the distribution $q(\mathbf{f}_m(\mathbf{X}_m))$, i.e., knowing the latents $\mathbf{f}_m(\mathbf{X}_m)$ would suffice to predict all remaining latent variables $\mathbf{f}(\mathbf{X})$. The conditional $p(\mathbf{f}(\mathbf{X})|\mathbf{f}_m(\mathbf{X}_m))$ can be explicitly computed for a given $\mathbf{X}$ and $\mathbf{X}_m$. Therefore, only $q(\mathbf{f}_m(\mathbf{X}_m))$ needs to be optimized in $q(\hat{\mathbf{f}}(\hat{\mathbf{X}}))$. Note that from here on, we stop explicitly denoting the dependence of the latent variables $\mathbf{f}$ on the input locations $\mathbf{X}$ to simplify notation.

We optimize the sparse distribution $q(\hat{\mathbf{f}})$ by minimizing the KL-divergence between the full GP modeling the underlying process $p(\hat{\mathbf{f}})$ and the sparse distribution $q(\hat{\mathbf{f}})$:

$$\mathcal{F}(q) = \mathrm{KL}(p(\hat{\mathbf{f}})||q(\hat{\mathbf{f}})) \tag{2.2}$$

But $p(\hat{\mathbf{f}})$ in the above formulation is only a prior on the stochastic process modeling the phenomenon of interest, and we need to provide it with labeled data $\mathbf{y}$, i.e., $p(\hat{\mathbf{f}}|\mathbf{y}))$ for it to represent the current state of the process. Including $\mathbf{y}$ has two key implications. First, it allows us to learn the GP hyperparameters (i.e., the kernel function parameters and data noise variance), thereby capturing the correlations in the stochastic process. Second, it biases our optimized sparse approximation to ensure that it captures the current realization of the stochastic process at the locations corresponding to the labels $\mathbf{y}$.

However, since our SP problem does not assume access to the ground truth labeled data, we can not include it in our formulation. We address this problem by introducing an uninformative training set with the labels $\mathbf{y}$ all set to zero, and the inputs are defined to be the locations within the boundaries of the environment. Such an

approach would still bias the sparse approximation to match the process within the confines of the monitoring environment. But we cannot use it to learn the hyperparameters. We address this issue by learning the hyperparameters from the sparse set of labeled data provided to us, as mentioned in the problem statement. We optimize the hyperparameters using the standard GP formulation and type II maximum likelihood [2].

We now address optimizing the sparse approximation q. The current formulation (Equation 2.2), which uses the forward KL-divergence, is computationally intractable to optimize if we include the labels [41]. Nonetheless, there are multiple alternative approaches to optimize the sparse approximation, such as variational inference and expectation proportion [3]. We leverage variational inference in this article, which gives us the following evidence lower bound (ELBO) as the new optimization objective:

$$\mathcal{F}(q) = \mathbb{E}_q \left[\log p(\hat{\mathbf{f}}, \mathbf{y}) \right] - \mathbb{E}_q \left[\log q(\hat{\mathbf{f}}) \right] . \tag{2.3}$$

Note that the above uses the reverse KL-divergence, and the training set labels $\mathbf{y}$ are all set to zero, with the corresponding inputs being the locations from within the monitoring environment. Substituting the full GP and the sparse approximation into the above ELBO gives us the following:

$$\begin{aligned} \mathcal{F}(q) &= \int q(\mathbf{f}_m) \log \frac{\mathcal{N}(\mathbf{y}|\boldsymbol{\alpha}, \sigma^2_{\text{noise}} I) p(\mathbf{f}_m)}{q(\mathbf{f}_m)} d\mathbf{f}_m \\ &\quad - \frac{1}{2\sigma^2_{\text{noise}}} Tr(\mathbf{K}_{ff} - \mathbf{Q}) \\ \boldsymbol{\alpha} &= \mathbb{E}[\mathbf{f} \mid \mathbf{f}_m] = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{f}_m \\ \mathbf{Q} &= \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn} . \end{aligned} \tag{2.4}$$

Here, σ_{noise} is the data noise, and the covariance matrix $\mathbf{K}$ subscripts indicate the variables used to compute it, with n indicating the n zero labeled points $\mathbf{X}$

corresponding to $\mathbf{y}$ and m indicating the m inducing points $\mathbf{X}_m$. By leveraging Jensen's inequality [3], we get the following optimal sparse distribution q^* as the solution to the ELBO:

$$q^*(\hat{\mathbf{f}}) = \mathcal{N}(\mathbf{f}_m|\sigma_{\text{noise}}^{-2}\mathbf{K}_{mm}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{mn}\mathbf{y}, \mathbf{K}_{mm}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{mm})\,, \tag{2.5}$$

where $\boldsymbol{\Sigma} = \mathbf{K}_{mm} + \sigma_{\text{noise}}^{-2}\mathbf{K}_{mn}\mathbf{K}_{nm}$. The input locations $\mathbf{X}_m$ corresponding to the auxiliary latents $\mathbf{f}_m$ used in the optimal sparse approximation can be optimized by maximizing the following expanded form of the ELBO:

$$\begin{aligned}\mathcal{F}(q) = \underbrace{\frac{n}{2}\log(2\pi)}_{\text{constant}} + \underbrace{\frac{1}{2}\mathbf{y}^\top(\mathbf{Q}_{nn} + \sigma_{\text{noise}}^2 I)^{-1}\mathbf{y}}_{\text{data fit}} + \\ \underbrace{\frac{1}{2}\log|\mathbf{Q}_{nn} + \sigma_{\text{noise}}^2 I|}_{\text{complexity term}} - \underbrace{\frac{1}{2\sigma_{\text{noise}}^2}Tr(\mathbf{K}_{nn} - \mathbf{Q}_{nn})}_{\text{trace term}}\,,\end{aligned} \tag{2.6}$$

where $\mathbf{Q}_{nn} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}$. The lower bound $\mathcal{F}$ has three key terms. The data fit term ensures that the training set labels are accurately predicted, which is disabled by setting the labels to zero. The complexity and trace terms are independent of the labels. The complexity term ensures that the inducing points (i.e., the input points $\mathbf{X}_m$ corresponding to the m auxiliary latent variables $\mathbf{f}_m$) are spread apart to ensure good coverage of the whole training set, and the trace term represents the sum of the variance of the conditional $p(\mathbf{f}|\mathbf{f}_m)$. When the trace term becomes zero, the m solution inducing points become a sufficient statistic for the n training samples, i.e., the m solution inducing points can make the same predictions as a GP with all the n samples in its training set.

The optimized $\mathbf{X}_m$ correspond to the solution sensor placement locations since the data collected at these locations can be used to recover the state of the full GP modeling the phenomenon of interest. Also, the inducing points can be optimized in continuous spaces using gradient-based approaches such as gradient descent and

Newton's method. In contrast to the non-differentiable SP approaches, the above formulation does not require one to discretize the environment, and is significantly faster to optimize. As such, the formulation scales well to large sensor placement problems.

Moreover, the above result can be viewed as a special case of sparse Gaussian processes (SGPs, [7, 8]) with the training set inputs corresponding to the locations within the data field being monitored and the labels all set to zero. In particular, our formulation closely follows that of the sparse variational free energy-based Gaussian process (SVGP); please refer to [8] for the full derivation of the above results.

Our formulation enables leveraging the vast SGP literature to address multiple variants of the SP problem. For instance, we can use stochastic gradient optimizable SGPs [42, 5] with our approach to address significantly large SP problems, i.e., environments that require a large number of sensor placements and have numerous obstacles that need to be avoided. Similarly, we can use spatiotemporal SGPs [1] with our approach to efficiently optimize sensor placements for spatiotemporally correlated environments. [11] conducted an in-depth analysis of the SVGP's lower bound, we can even leverage these findings to understand how to better optimize the inducing points in our formulation. Additionally, this formulation enables us to bound the KL divergence between the sparse approximation and the ground truth full GP modeling the underlying phenomenon in the environment:

Theorem 1. [43] Suppose N training inputs are drawn i.i.d according to input density $p(\mathbf{x})$, and $k(\mathbf{x}, \mathbf{x}) < v$ for all $\mathbf{x} \in \mathbf{X}$. Sample M inducing points from the training data with the probability assigned to any set of size M equal to the probability assigned to the corresponding subset by an ϵ k-Determinantal Point Process with k $= M$. With probability at least $1 - \delta$,

$$KL(Q||\hat{P}) \leq \frac{C(M+1) + 2Nv\epsilon}{2\sigma_n^2\delta}\left(1 + \frac{||\mathbf{y}||_2^2}{\sigma_n^2}\right)$$

where $C = N \sum_{m=M+1}^{\infty} \lambda_m$, λ_m are the eigenvalues of the integral operator $\mathcal{K}$ for kernel k and $p(\mathbf{x})$.

Here, k represents the k-value in the k-determinantal point process, and k represents the kernel function. In our sensor placement problem, Q is equivalent to the sparse approximation that can be used to predict the state of the whole environment from sensor data collected at the inducing points, and $\hat{P}$ is the ground truth GP that senses every location in the environment. The theorem also suggests an asymptotic convergence guarantee, i.e., as the number of sensing locations increases, the probability of the sparse approximation becoming exact approaches one.

Please refer to Appendix A.1 for additional theoretical analysis of the approach, which details our derivation to show that our formulation is not submodular and the conditions under which it behaves similarly.

2.4.2 Continuous-SGP: Continuous Space Solutions

Next, we detail Algorithm 1, which leverages the formulation above to address the SP problem in continuous spaces. Given an environment, we first sample random unlabeled locations within the boundaries of the monitoring regions. These unlabeled locations $\mathbf{X}$ are used as the training set inputs, and their labels are all set to zero.

Next, we sample unlabeled locations to initialize the inducing points $\mathbf{X}_m$ of the SGP used to solve the SP problem. As per the findings of [43], the initial inducing points' locations significantly affect the quality of the sparse approximation that can be obtained with SGPs. We recommend using methods such as the conditional variance [43] or K-means [3] on the randomly sampled training dataset to initialize the inducing points.

We use the sampled unlabeled training set and the inducing points to initialize an SGP. This article uses the spare variational free energy-based Gaussian process (SVGP) as it is the SGP counterpart to our formulation. However, as mentioned in the previous subsection, any SGP approach can be used in our algorithm to address

Algorithm 1: Continuous-SGP approach for obtaining sensor placements in continuous spaces. Here, θ are the hyperparameters learned from either historical data or expert knowledge, Φ is a random distribution defined within the boundaries of the environment $\mathcal{V}$, s is the number of required sensors, n is the number of random unlabeled locations used to train the SGP, and γ is the SGP learning rate.

Input: $\theta, \mathcal{V}, \Phi$, s, n, γ
Output: Sensor placements $\mathcal{A} \in \mathcal{V}$, where $|\mathcal{A}| = s$
$\mathbf{X} \sim \Phi(\mathcal{V})$ // Draw n unlabeled locations
$\mathbf{X}_m \sim \Phi(\mathcal{V})$ // Draw s inducing point locations
`/* Initialize the SGP with zero mean and zero labeled data */`
$\varphi = \mathcal{SGP}(\text{mean} = 0, \theta; \mathbf{X}, \mathbf{y} = \mathbf{0}, \mathbf{X}_m)$
`/* Optimize the inducing points` $\mathbf{X}_m$ `by maximizing the objective function` $\mathcal{F}$ `of the SGP` φ `using gradient descent with a learning rate of` γ `*/`
Loop until *convergence* :
    $\mathbf{X}_m \leftarrow \mathbf{X}_m + \gamma \nabla \mathcal{F}(\mathbf{X}_m)$
return $\mathbf{X}_m$

the SP problem.

We then optimize the inducing point locations of the SGP via the complexity and trace terms of the lower bound $\mathcal{F}$; the data fit term is disabled by the zero labeled training set. Since the lower bound $\mathcal{F}$ is fully differentiable, we can use gradient-based approaches to efficiently optimize the inducing points in continuous spaces. This gives us the optimized inducing points, which, in turn, represent the solution sensor placement locations.

2.4.3 Greedy-SGP: Greedy Discrete Space Solutions

Now consider the case when we want to limit the solution of the SP problem to a discrete set of candidate locations, either a subset of the training points or any other arbitrary set of points. In this case, we can generalize the inducing points selection approach for SGPs outlined in [8] to handle non-differentiable data domains.

The approach (Algorithm 2) entails sequentially selecting the inducing points $\mathbf{X}_m$ from the candidate set $\mathcal{S}$ using a greedy approach (Equation 2.7). It considers the increment in the SVGP's optimization bound $\mathcal{F}$ as the maximization criteria. In each

Algorithm 2: Greedy-SGP approach for obtaining sensor placements in discrete environments (i.e., sensor placements limited to a given set of candidate sensor locations) using a greedy selection approach. Here, θ are the hyperparameters learned from either historical data or expert knowledge, $\mathcal{S}$ is the set of candidate sensor placement locations, Φ is a random distribution defined within the boundaries of the environment $\mathcal{V}$, and s is the number of required sensors, n is the number of random unlabeled locations used to train the SGP.

Input: $\theta, \mathcal{V}, \mathcal{S}, \Phi,\ s,\ n$
Output: Sensor placements $\mathcal{A} \subset \mathcal{S}$, where $|\mathcal{A}| = s$

$\mathbf{X} \sim \Phi(\mathcal{V})$ // Draw n unlabeled locations
`/* Initialize the SGP with zero mean and zero labeled data */`
$\varphi = \mathcal{SGP}(\text{mean} = 0, \theta; \mathbf{X}, \mathbf{y} = \mathbf{0})$
`/* Sequentially select each of the solution inducing point locations using the criteria defined in terms of the optimization bound` $\mathcal{F}$ `of the SGP` φ `*/`
$\mathbf{X}_m = \{\emptyset\}$
repeat
  $\mathbf{x}_m^* = \arg\max_{\mathbf{x} \in \mathcal{S} \setminus \mathbf{X}_m} \mathcal{F}(\mathbf{X}_m \cup \{\mathbf{x}\}) - \mathcal{F}(\mathbf{X}_m)$
  $\mathbf{X}_m \leftarrow \mathbf{X}_m \cup \mathbf{x}_m^*$
until $|\mathbf{X}_m| = s$;
return $\mathbf{X}_m$

iteration, we select the point $\mathbf{x}$ that results in the largest increment in the SVGP's bound $\mathcal{F}$ upon being added to the current inducing points set $\mathbf{X}_m$:

$$\mathbf{X}_m \leftarrow \mathbf{X}_m \cup \{\arg\max_{\mathbf{x} \in \mathcal{S} \setminus \mathbf{X}_m} \mathcal{F}(\mathbf{X}_m \cup \{\mathbf{x}\}) - \mathcal{F}(\mathbf{X}_m)\}\,. \tag{2.7}$$

Here $\mathbf{X}_m$ is the set of inducing points/sensing locations, and $\mathcal{S} \setminus \mathbf{X}_m$ is the set of remaining candidate locations after excluding the current inducing points set $\mathbf{X}_m$.

Comparison with Mutual Information: The Greedy-SGP approach has a few interesting similarities to the mutual information (MI) based sensor placement approach by [13]. The MI approach uses a full GP to evaluate MI between the sensing locations and the rest of the environment to be monitored. The MI-based criteria shown below was used to greedily select sensing locations:

$$MI(\mathbf{X}_m \cup \{\mathbf{x}\}) - MI(\mathbf{X}_m) = H(\mathbf{x}|\mathbf{X}_m) - H(\mathbf{x}|\mathcal{S} \setminus \mathbf{X}_m)\,, \tag{2.8}$$

where $\mathbf{X}_m$ is the set of selected sensing locations, and $\mathcal{S}\backslash\mathbf{X}_m$ is the set of all candidate locations in the environment excluding the current sensor locations $\mathbf{X}_m$. A Gaussian process (GP) with known kernel parameters was used to evaluate the entropy terms. The SVGP's optimization bound-based selection criterion to obtain discrete solutions using the greedy algorithm is equivalent to maximizing the following:

$$\Delta\mathcal{F} = \mathrm{KL}(q(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y})) - \mathrm{KL}(p(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y}))\,. \tag{2.9}$$

The first KL term measures the divergence between the sparse distribution q over f_i (the latent variable corresponding to $\mathbf{x}$) given the latents of the inducing points set $\mathbf{X}_m$, and the exact conditional given the training set labels $\mathbf{y}$ (the conditional uses the training set inputs $\mathbf{X}$ as well). The second term acts as a normalization term that measures the divergence between the exact conditional over f_i given the latents of the inducing points and the same given the training labels.

A key difference between the Greedy-SGP and the MI approach is that we use efficient cross-entropy (in the KL terms) to account for the whole environment. In contrast, the MI approach uses the computationally expensive entropy term $H(\mathbf{x}|\mathcal{S}\backslash\mathbf{X}_m)$. However, the overall formulation of both approaches is similar. We validate this empirically in the experiments section. Please refer to Appendix A.1 for the derivation of Equation 2.9.

2.4.4 Discrete-SGP: Gradient-based Discrete Space Solutions

The problem with any greedy selection algorithm is its inherent sequential selection procedure. Given any sequentially optimized solution, it may be possible to improve the solution by replacing one or more of the solution locations with different locations from the candidate set. Thus, selecting the sensing locations sequentially restricts the solution placements to a subset of the solution space. We could find a better solution

by simultaneously optimizing all the sensing locations as such an approach would account for their combined effect instead of considering only the incremental effect of the solution locations in each selection iteration.

Algorithm 3: Discrete-SGP approach for obtaining sensor placements in discrete environments (i.e., sensor placements limited to a given set of candidate sensor locations) using gradient descent. Here, θ are the hyperparameters learned from either historical data or expert knowledge, $\mathcal{S}$ is the set of candidate sensor placement locations, Φ is a random distribution defined within the boundaries of the environment $\mathcal{V}$, s is the number of required sensors, n is the number of random unlabeled locations used to train the SGP, and γ is the SGP learning rate.

Input: $\theta, \mathcal{V}, \mathcal{S}, \Phi, s, n, \gamma$
Output: Sensor placements $\mathcal{A} \subset \mathcal{S}$, where $|\mathcal{A}| = s$
/* Get the s continuous space sensor placements using the gradient based approach Continuous-SGP (Algorithm 1) */
$\mathbf{X}_m = \text{Continuous-SGP}(\theta, \mathcal{V}, \Phi, s, n, \gamma)$
// Compute pairwise L_2 distances
$\mathbf{C} = \mathbf{0}^{|\mathbf{X}_m| \times |\mathcal{S}|}$
for $i \leftarrow 0$ **to** $|\mathbf{X}_m|$ **do**
  for $j \leftarrow 0$ **to** $|\mathcal{S}|$ **do**
    $\mathbf{C}[i][j] \leftarrow ||\mathbf{X}_m[i] - \mathcal{S}[j]||_2$
/* Solve the assignment problem $\mathcal{H}$ to assign the s continuous space inducing points $\mathbf{X}_m$ to locations in the candidate set $\mathcal{S}$ */
$A = \mathcal{H}(\mathbf{C})$
/* Use the assignments A to index the candidate set $\mathcal{S}$ and get the discrete space solution */
$\mathbf{X}^*_m = \mathcal{S}[A]$
return $\mathbf{X}^*_m$

Our approach (Algorithm 3) to this problem is to simultaneously optimize all the inducing points in the continuous input space using gradient descent and then map the solution to the discrete candidate solution space $\mathcal{S}$. We can map the continuous space solutions to discrete sets by treating the mapping problem as an assignment problem [44], i.e., as a weighted bipartite matching problem. The assignment problem requires one to find the minimal cost matching of a set of items to another set of items given their pairwise costs. We compute the pairwise Euclidean distances between the

continuous space inducing points and the discrete space candidate set locations $\mathcal{S}$. The distances are then used as the costs in an assignment problem. One could even use covariances that are appropriately transformed, instead of distances, in the mapping operation to account for the correlations in the environment.

The solution of the assignment problem gives us points in the discrete candidate set closest to the continuous space solution set. Such a solution could be superior to the greedy solution since the points in the continuous space solution set are simultaneously optimized using gradient descent instead of being sequentially selected. Although the gradient-based solution could get stuck in a local optimum, in our experiments, we found that the gradient-based discrete solutions are on par or better than the greedy solutions while being substantially faster to optimize.

2.5 Experiments

We demonstrate our methods on four datasets—Intel lab temperature [45], precipitation [46], soil moisture [47], and ROMS ocean salinity [48]. The datasets are representative of real-world sensor placement problems and some of these have been previously used as benchmarks [13]. We used an RBF kernel [2] in these experiments.

The Intel lab temperature dataset contains indoor temperature data collected from 54 sensors deployed in the Intel Berkeley Research lab. The precipitation dataset contains daily precipitation data from 167 sensors around Oregon, U.S.A, in 1994. The US soil moisture dataset contains moisture readings from the continental USA, and the ROMS dataset contains salinity data from the Southern California Bight region. We uniform sampled 150 candidate sensor placement locations in the soil and salinity datasets.

For each dataset, we used a small portion of the data to learn the kernel parameters. and used a Gaussian process (GP) to reconstruct the data field in the environment from each method's solution placements. The GP was initialized with the learned kernel function, and the solution sensing locations and their corresponding ground

truth labels were used as the training set in the GP. We evaluated our data field reconstructions using the root-mean-square error (RMSE).

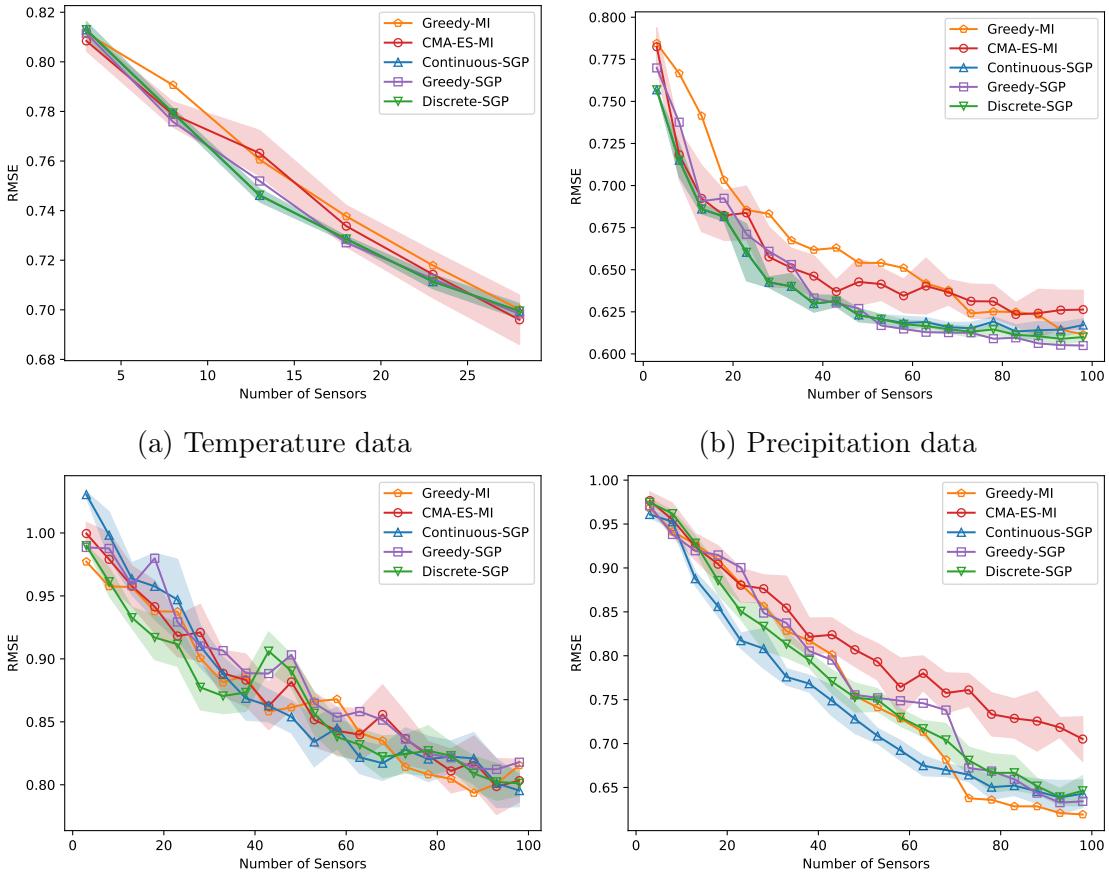
We trained all GPs (and SGPs) with a learning rate of $1e-2$ for a maximum of 3000 iterations using the Adam optimizer [49]. We used the GPflow Python library [50] for all our GP implementations, and the apricot Python library [51] for the greedy selection algorithm. All our experiments were executed on a Dell workstation with an Intel(R) Xeon(R) W-2265 CPU and 128 GB RAM. We ran our experiments using Python 3.8.10.

2.5.1 Discrete and Continuous-Space Sensor Placement

We benchmarked our approaches—Continuous-SGP (Section 2.4.2), Greedy-SGP (Section 2.4.3), and Discrete-SGP (Section 2.4.4). We also evaluated the performance of the approach in [13], which maximizes mutual information (MI) using the greedy algorithm (Greedy-MI) in discrete environments, and we used the covariance matrix adaptation evolution strategy (CMA-ES) to maximize MI (CMA-ES-MI) as another baseline as it can handle continuous environments. We chose these baselines as they have also been show to perform well on sensor placement problems[1]. All methods considered the sparsely labeled sensor placement problem, i.e., only a small portion of labeled data was available to learn the kernel parameters.

We computed the solution sensor placements for 3 to 100 sensors (in increments of 5) for all the datasets, except for the Intel dataset, which was tested for up to 30 placement locations since it has only 54 placement locations in total. The experiments were repeated 10 times and we report the mean and standard deviation of the RMSE and runtime results in Figures 2.2 and 2.3. We see that our approaches' RMSE results are consistently on par or better that the baseline Greedy-MI and CMA-ES-MI approaches.

[1]We also generated results using Bayesian Optimization (BO) as another baseline. However the results were suboptimal. Please refer to Appendix A.2 for the BO results.



(a) Temperature data

(b) Precipitation data

(c) Soil data

(d) Salinity data

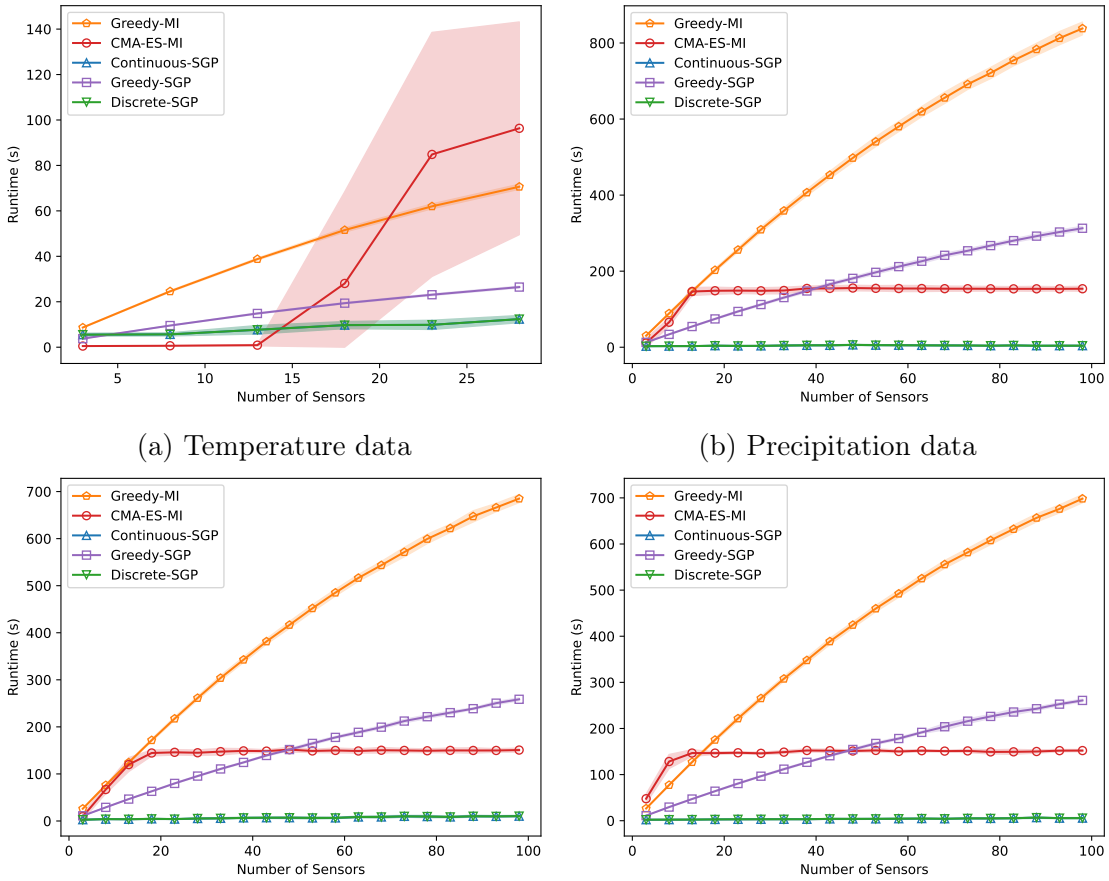
Figure 2.2: The mean and standard deviation of the RMSE vs number of sensors for the Intel, precipitation, soil, and salinity datasets (lower is better).

Moreover, our approaches are substantially faster than the baseline approaches. Our Continuous-SGP approach is up to 6 times faster than the baselines in the temperature dataset, up to 50 times faster in the precipitation dataset, and up to 43 times faster in the soil and salinity datasets. Our SGP-based approaches select sensing locations that reduce the RMSE by maximizing the SGP's ELBO, which requires inverting only an $m \times m$ covariance matrix ($m \ll |\mathcal{S}|$, where $|\mathcal{S}|$ is the number of candidate locations). In contrast, both the baselines maximize MI, which requires inverting up to an $|\mathcal{S}| \times |\mathcal{S}|$ covariance matrix to place each sensor, which takes $\mathcal{O}(|\mathcal{S}|^3)$ time. As such, the computational cost difference is further exacerbated in the precipitation, soil, and salinity datasets, which have three times as many candidate locations

as the temperature data.

Also, the Continuous-SGP method consistently generates high quality results in our experiments; which is consistent with the findings of [11], who showed that SVGP's are able to recover the full GP posterior in regression tasks. Solving the assignment problem in our Discrete-SGP approach to map the Continuous-SGP solution to the discrete candidate set incurs a one-time $\mathcal{O}(m^3)$ computation time that is negligible. Therefore our gradient-based approaches—Continuous-SGP and Discrete-SGP—converge at almost the same rate. Yet the Discrete-SGP retains the solution quality of the Continuous-SGP solution.

The labeled data locations in the soil and salinity datasets used to train our kernel



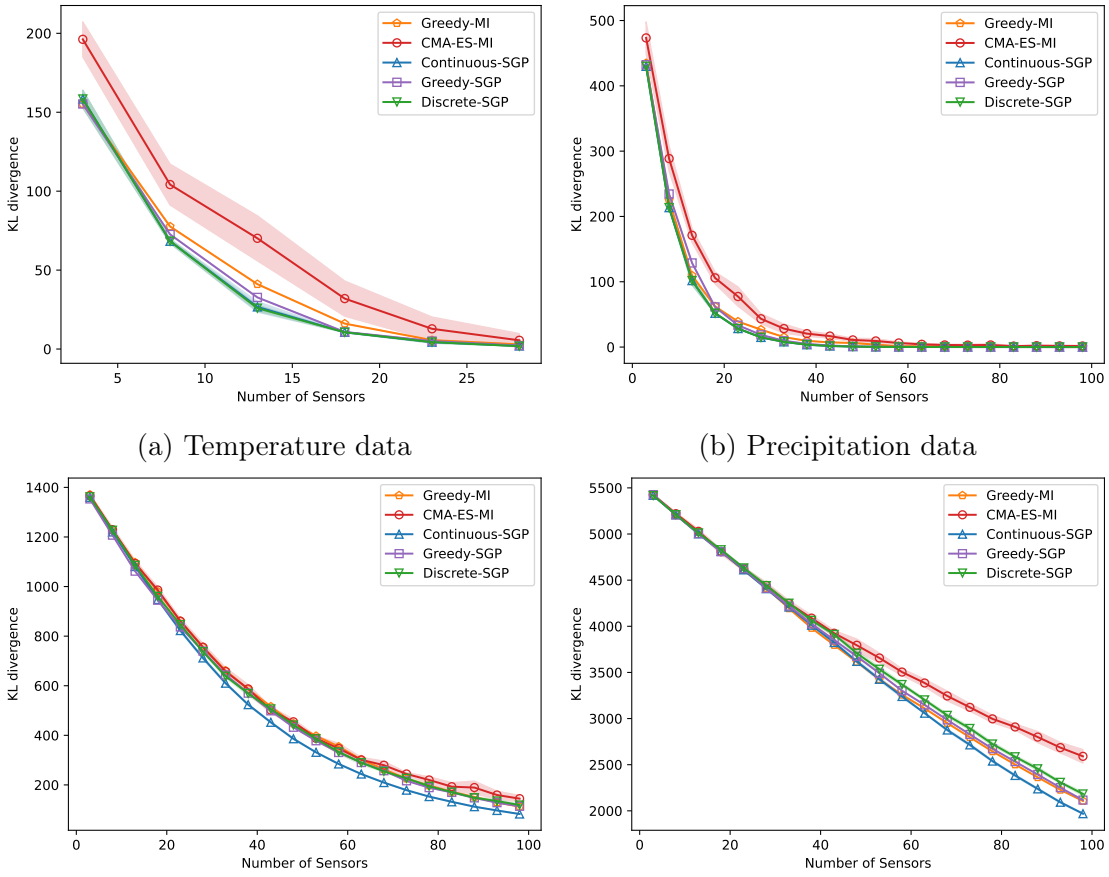
(a) Temperature data

(b) Precipitation data

(c) Soil data

(d) Salinity data

Figure 2.3: The mean and standard deviation of the Runtime vs number of sensors for the Intel, precipitation, soil, and salinity datasets (lower is better).



(a) Temperature data

(b) Precipitation data

(c) Soil data

(d) Salinity data

Figure 2.4: KL divergence between the SGP posterior and the true posterior vs number of sensors for the temperature, precipitation, soil, and salinity datasets (lower is better).

function were not aligned with our candidate locations for the discrete approaches. We chose this setup to demonstrate that we can learn the kernel parameters even if the data is not aligned with the candidate locations, or is from a different environment altogether.

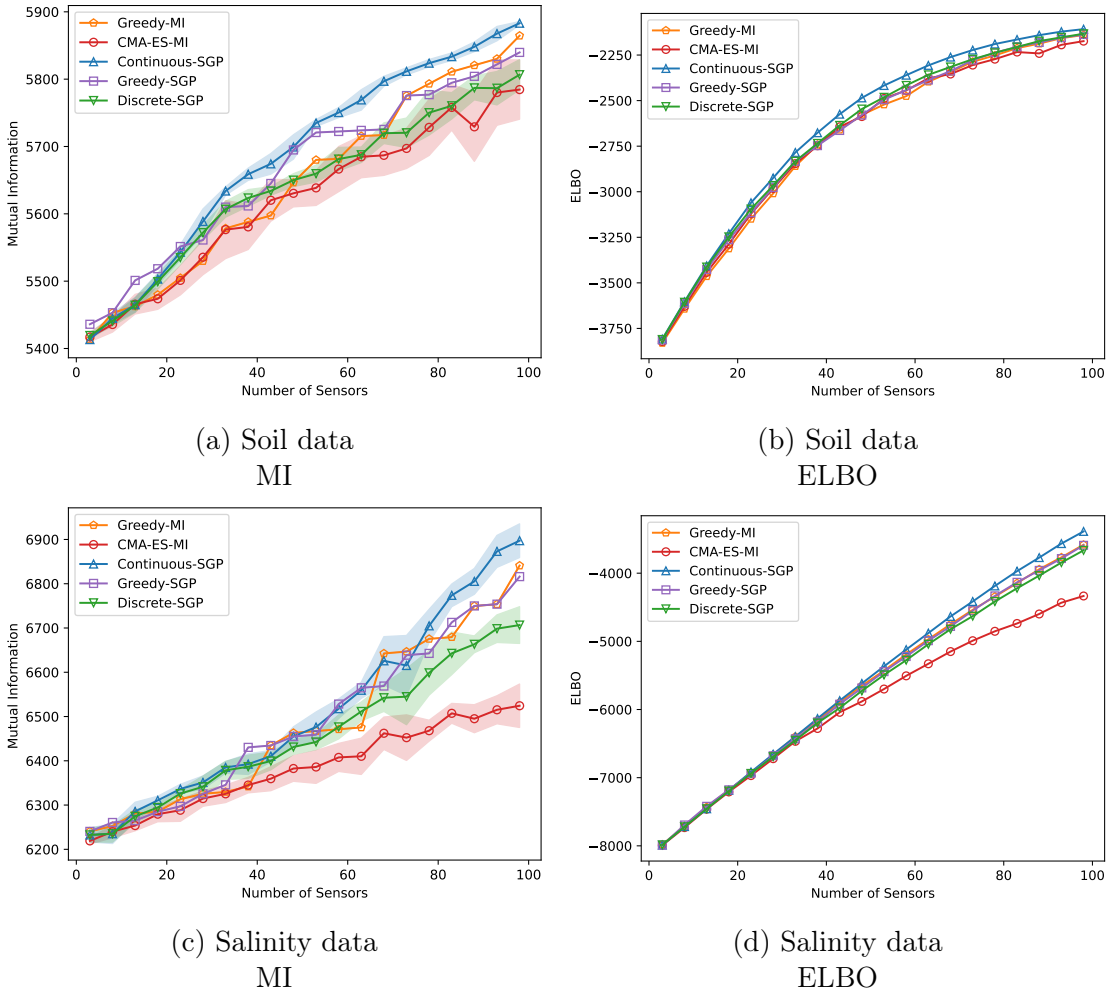
In Figure 2.4, we show how each of the methods presented in this chapter—Greedy-MI, CMA-ES-MI, Continuous-SGP, Greedy-SGP, and Discrete-SGP—perform on the KL divergence between the SGP posterior and the true posterior measured using the approach presented in [12]. We see the our approaches consistently perform on par or better than the baselines.

2.5.2 Comparison with Mutual Information

In Figure 2.5 we show the MI and SVGP's lower bound (ELBO) between the solution placements and the environments (2500 uniformly sampled locations). The label dependent data fit term in the ELBO was disabled to generate the shown results. Although the ELBO values differ from MI, they closely approximate the relative trends of the MI values. This validates our claim that the SVGP's ELBO behaves similarly to MI while being significantly cheaper to compute.

2.5.3 Spatiotemporal Sensor Placement

We demonstrate our approach's scalability to large spatiotemporal data fields by finding placements for 500 ozone concentration sensors across the planet. Note that the environment is the surface of a sphere in this example. We used a spatiotemporal-sparse variational Gaussian process (ST-SVGP) [1] as it allows us to efficiently model spatiotemporal correlations in the data with time complexity linear in the number of time steps in the training set. We used Matern 3/2 kernels [2] to model the spatial and temporal correlations. All the model parameters were optimized with a learning rate of 0.01, and the parameters were optimized using the Adam optimizer [49]. The ST-SVGP was trained on the first six months of the monthly ozone data from 2018 [52],



(a) Soil data MI

(b) Soil data ELBO

(c) Salinity data MI

(d) Salinity data ELBO

Figure 2.5: Comparison of the MI and SVGP's lower bound (ELBO) for the soil and salinity datasets. The mean and standard deviation of the MI vs number of sensors (a), (c) and SVGP's lower bound (ELBO) vs number of sensors (b), (d).

and we used a subset of 1040 uniformly distributed locations in the dataset as the training set and 100 inducing points to learn the kernel parameters. The learned kernel function was then used in our sensor placement approach—Continuous-SGP (Algorithm 1)—to obtain the 500 solution placements shown below. Note that the solution sensor placements are spatially fixed to monitor the spatiotemporal data.

The solution placements are relatively uniformly distributed over the planet. This is because we used a stationary kernel function. However, in a real-world scenario, using a non-stationary kernel would give us even more informative sensing locations

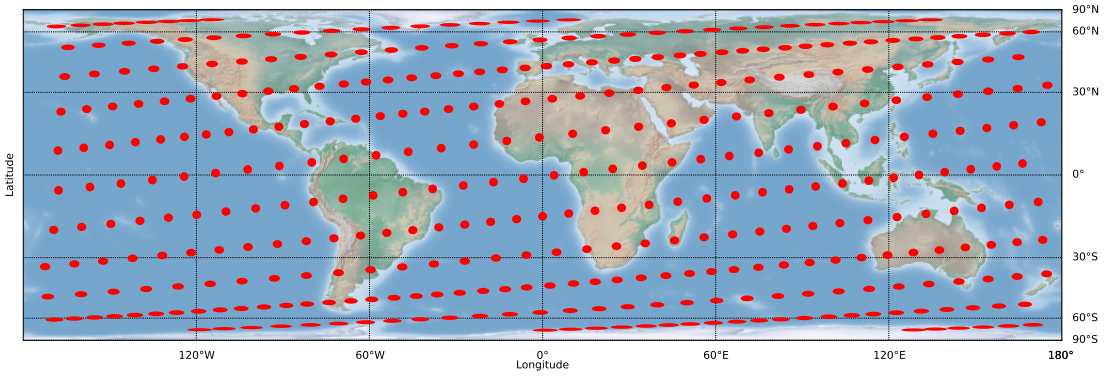


Figure 2.6: Placements for 500 sensors generated using the Continuous-SGP approach with an ST-SVGP [1]. The red points are the sensor placements projected onto the 2D map using cylindrical equal-area projection.

that can further leverage the non-stationary nature of the environment.

2.5.4 Obstacle Avoidance

We handle obstacles in the environment by building an appropriate training dataset for the SGP. We remove the random samples in the SGP training set at locations in the interior of obstacles. Therefore the resulting training set has samples only in obstacle-free regions. Training an SGP on such data would result in inducing points that avoid the obstacles since placing the inducing points at locations with obstacles would not increase the likelihood of the training data used to optimize the SGP. Our obstacle avoidance approach is best suited for relatively large obstacles.

We now present our solution sensor placements in an environment with multiple obstacles (Figure 2.7). We trained the SGP using gradient descent on randomly sampled points in the environment where there were no obstacles and set all labels to zero. As we can see, the solution placements are well-spaced to ensure that the same information is not repeatedly collected. Also, our solution placements perfectly avoid the obstacles.

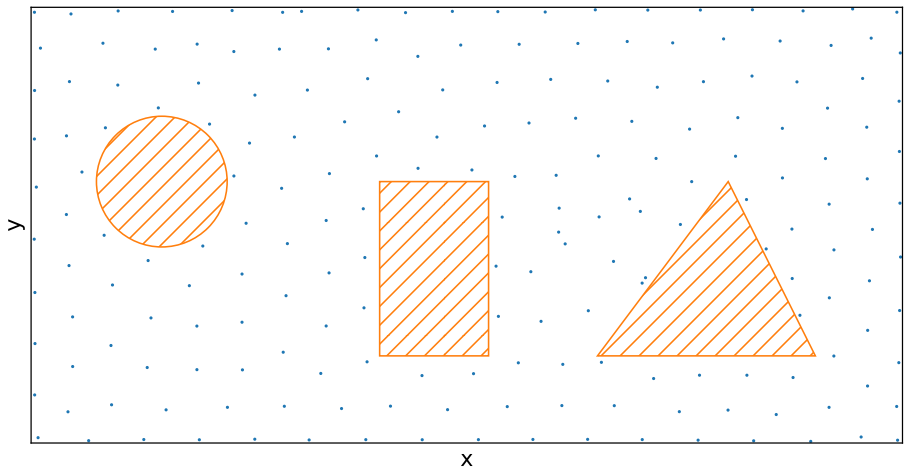


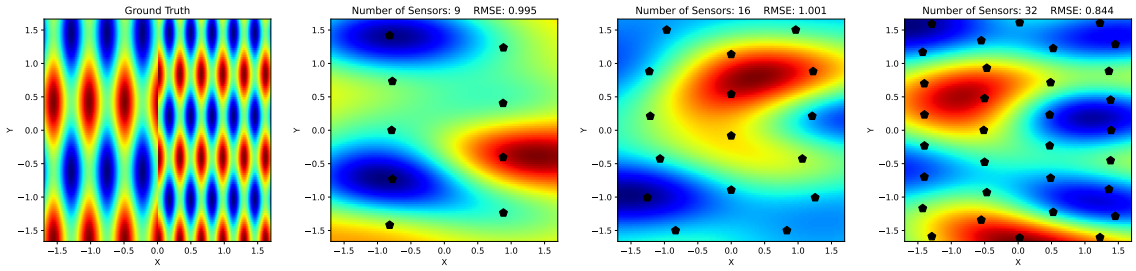
Figure 2.7: Placements for 200 sensors generated using the SGP approach. The hatched orange polygons represent obstacles in the environment and the blue points represent the solution sensor placements.

2.5.5 Non-Stationarity Kernels

Based on the above experiments, when using stationary kernels, the solution sensor placements appear to be uniformly distributed. Indeed, the benefit of the method is in determining the density of the sensing locations and avoiding the obstacles. This is true for approaches that maximize MI [13, 36, 37] and our SGP approach, which is also an efficient method to approximately maximize MI. One can also use a faster uniform sampling technique such as Latin hypercube sampling [53] to achieve similar results if the ideal sensing density is known and there are no obstacles in the environment.

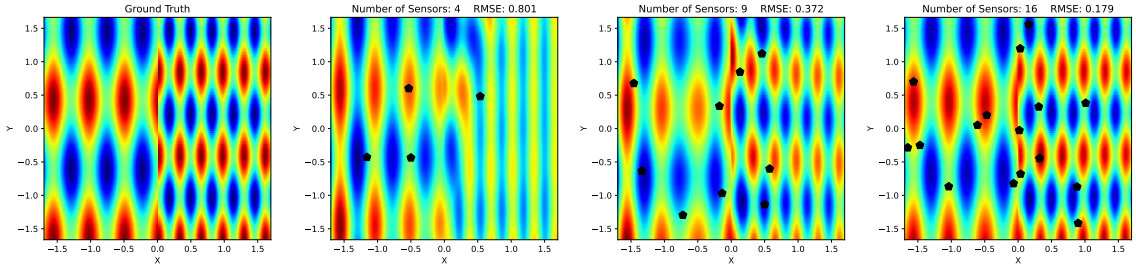
However, when considering non-stationary environments, the key advantage of GP-based methods (including our SGP approach) becomes apparent. We generated a non-stationary data field (elevation data), in which the left half varies at a lower frequency than the right half. Figure 2.8 shows the sensor placement results generated using the SGP approach with a stationary RBF kernel [2]. The ground truth data from the sensor placements was used to generate a dense reconstruction of the non-stationary environment (20,000 data points). As we can see, the method performs poorly at

reconstructing the ground truth data.



(a) (b) (c) (d)

Figure 2.8: A non-stationary environment. (a) Ground truth. Reconstructions from the Continuous-SGP solutions with a stationary RBF kernel function for (b) 9, (c) 16, and (d) 32 sensing locations. The black pentagons represent the solution placements.



(a) (b) (c) (d)

Figure 2.9: A non-stationary environment. (a) Ground truth. Reconstructions from the Continuous-SGP solutions with a neural kernel for (b) 4, (c) 9 and, (d) 16 sensing locations. The black pentagons represent the solution placements.

We can address the above issue by leveraging a non-stationary kernel. We trained a neural kernel [54] to learn the correlations in the environment. We used three mixture components in the neural kernel function and parameterized each constituent neural network as a two-layer multilayer perceptron [3] with four hidden units each. The training data consisted of 1250 grid-sampled labeled data from the non-stationary environment. The neural kernel function parameters were optimized using a Gaussian process (GP) [2] trained with type-II maximum likelihood [3]. We then used the neural kernel function in our Continuous-SGP approach (Algorithm 1) to generate sensor placements for 4, 9, and 16 sensing locations. The results with the non-stationary neural kernel are shown in Figure 2.9 for 4, 9, and 16 sensor placements.

As we can see in Figure 2.9, the sensor placements are no longer uniformly distributed. Instead, they are biased towards strategic locations that are crucial for accurately estimating the data field. We see multiple sensors in the middle, where the frequency of the variations (elevation) change, thereby accurately capturing the underlying data field.

2.6 Conclusion

We addressed the sensor placement problem for monitoring correlated data. We formulated the problem as a regression problem using SGPs and showed that training SGPs on unlabeled data gives us ideal sensor placements in continuous spaces, thereby opening up the vast GP and SGP literature to the sensor placement problem and its variants involving constraints, non-point sensors, etc. The method also enables us to efficiently handle sensor placement in 3D spaces, spatio-temporally correlated spaces, and leverage the convergence rate proofs of SGPs. Furthermore, we presented an approach that uses the assignment problem to map the continuous domain solutions to discrete domains efficiently, giving us computationally efficient discrete solutions compared to the greedy approach. Our experiments on four real-world datasets demonstrated that our approaches result in both mutual information and reconstruction quality being on par or better than the existing approaches while substantially reducing the computation time.

A key advantage of our approach is its differentiability with respect to the sensing locations, which allows us to incorporate it into deep neural networks optimized for more complex downstream tasks. We leverage this differentiability property in concurrent work to generalize our sensor placement approach to robotic informative path planning. Since our approach, and more generally GP-based approaches, rely on accurate kernel function parameters, we aim to develop online approaches to address this in our future work.

CHAPTER 3: Informative Path Planning in Continuous and Discrete Environments for Data Field Estimation

3.1 Introduction

Environmental monitoring problems require estimating the current state of phenomena, such as temperature, precipitation, ozone concentration, soil chemistry, ocean salinity, and fugitive gas density ([55, 56, 15, 17]). These problems are closely related to the informative path planning (IPP) problem ([55, 57]) since it is often the case that we have limited resources and, therefore, must strategically determine the regions from which to collect data and the order in which to visit the regions to efficiently and accurately estimate the state of the environment.

The IPP problem has been studied in numerous scenarios: [58] developed IPP for persistent ocean monitoring with underwater gliders, [19] studied IPP for information gathering on three-dimensional mesh surfaces for inspection tasks, [17] presented an IPP approach for localizing gas sources in oil fields, and [59] used IPP for active learning in aerial semantic mapping.

Most IPP approaches implicitly assume that the environment is correlated ([55, 58, 57, 60, 36, 56, 37]). Similarly, we consider IPP problems for environments that are correlated either spatially or spatio-temporally and present an efficient approach that leverages such correlations.

Existing discrete optimization based IPP methods have discretization requirements that limit them to relatively small problems ([55, 57, 56]), making them infeasible for large spatio-temporal environments. Additionally, incorporating routing constraints, such as a distance budget and limits on the robot's velocity and acceleration, significantly increase the problem size when using discrete optimization.

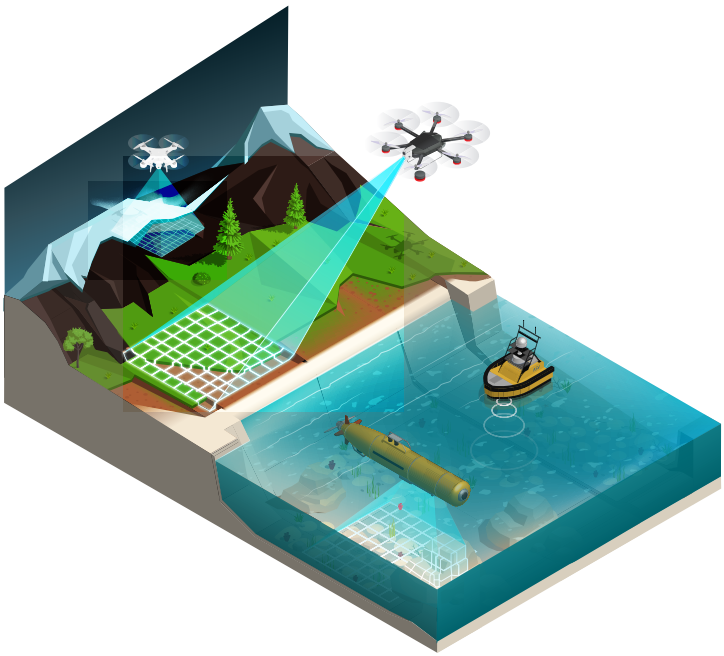


Figure 3.1: Illustration of informative path planning using multiple robots.

Furthermore, modeling informative paths in continuous domains with potentially continuous sensing robots is a non-trivial problem. The problem is usually addressed using optimization methods such as rapidly-exploring random trees (RRT), genetic algorithms, or Bayesian optimization ([60, 36, 37]). These methods select sensing locations that maximize mutual information (MI) computed using Gaussian processes [13]. But some of these optimization methods are computationally expensive and rely on computing MI, which is also expensive ($\mathcal{O}(n^3)$, where n is the discretized environment size). A few approaches have even considered multi-robot IPP ([61, 36]) but they are also inherently limited by the scalability issues of prior IPP approaches.

Motivated by the above limitations of prior IPP approaches, we present a method that can efficiently generate both discrete and continuous sensing paths, accommodate constraints such as a distance budget and velocity limits, handle point sensors and non-point FoV sensors, and handle both single and multi-robot IPP problems. Our approach leverages gradient descent optimizable sparse Gaussian processes to solve the IPP problem, making it significantly faster compared to prior approaches and

scalable to large IPP problems.

3.2 Multi-Robot Informative Path Planning Problem

We consider a spatially (or spatiotemporally) correlated stochastic process over an environment $\mathcal{V} \subseteq \mathbb{R}^d$ representing a phenomenon such as temperature. We have r robots and must find the set $\mathcal{P}$ of r paths, one for each robot, so that the data from the phenomenon collected at these locations is sufficient to accurately estimate the phenomenon at every location in the environment. We use the root-mean-square error (RMSE) of the estimates as the measure of accuracy. Since we cannot directly minimize the RMSE, we formulate this problem as one where we want to find the paths $\mathcal{P}$ that maximize the information I (i.e., any function that is a good proxy for accuracy and can be computed without the ground truth labels). Moreover, we also consider constraints $\mathbf{C}$ such as distance budget and velocity limits on the paths:

$$\begin{aligned} \mathcal{P}^* = \underset{\{\mathcal{P}_i \in \psi, i=1,\dots,r\}}{\arg\max} \quad & I(\cup_{i=1}^{r} \text{SAMPLE}(\mathcal{P}_i)), \\ \text{s.t. } & \text{Constraints}(\mathcal{P}_{i=1,\dots,r}) \leq \mathbf{C} \end{aligned} \tag{3.1}$$

Here ψ is the space of paths contained within the environment $\mathcal{V}$, and the SAMPLE function returns the sensing points along a path $\mathcal{P}_i$. When considering discrete sensing robots, each path is constrained to have only s sensing locations. In a continuous sensing model, the SAMPLE function returns all the points along the path, which are used to compute the integral of the information collected along the path. In addition, we also consider point sensors such as temperature probes, and non-point sensors that can have any field-of-view (FoV) shape such as a thermal vision camera with a rectangular FoV.

3.3 Related Work

The Informative Path Planning (IPP) Problem is known to be NP-hard [62]. Therefore, only suboptimal solutions can be found for most real-world problems. Numerous

IPP methods select utility functions that are submodular ([63, 61, 64, 57]). Submodular functions have a diminishing returns property that can be leveraged to get good approximation guarantees even when optimized using greedy algorithms.

Many IPP approaches use mutual information (MI), an information metric that is submodular [13], as the optimization objective. The methods compute MI using Gaussian processes with known kernel parameters. But MI requires one to discretize the environment, thereby limiting the precision with which the sensing locations can be selected. Also, MI is computationally expensive ($\mathcal{O}(n^3)$, where n is the number of locations in the discretized environment). Singh et al. [65] proposed a recursive-greedy algorithm that maximized MI. The approach addressed both single and multi-robot IPP. Ma et al. [56] solved the IPP problem by maximizing MI using dynamic programming and used an online variant of sparse Gaussian processes for efficiently learning the model parameters. Bottarelli et al. [66] developed active learning-based IPP algorithms with a complexity of $\mathcal{O}(|D|^5)$, where D is the discretized data collection space.

Hollinger and Sukhatme [60] presented IPP algorithms for continuous spaces that maximized MI using rapidly-exploring random trees (RRT) and derived asymptotically optimal guarantees. Miller et al. [67] addressed continuous-space IPP with known utility functions using an ergodic control algorithm. Hitz et al. [36] and Popovic et al. [68] developed IPP approaches that could optimize the sensing locations in continuous spaces by optimizing any utility function. The approaches used a B-spline to parametrize a path and maximized the utility function (mutual information) using a genetic algorithm. Some authors even leveraged Bayesian optimization [37, 69] to find informative paths in continuous spaces. However, similar to discrete optimization and genetic algorithm based approaches, the method was computationally expensive and limited the approach's scalability. Mishra et al. [70] addressed IPP using dynamic programming, but the approach utilized variance as

the information metric, which can be computed quickly but is not as informative as mutual information.

A closely related problem is the correlated orienteering problem (COP), in which one has to plan a path that maximizes the information gain in a correlated environment while restricting the path to a given distance budget [71]. Agarwal and Akella [72] addressed COP for both point locations and 1D features using quadratic programming.

Recently, Rückin et al. [73] leveraged deep reinforcement learning (DRL) to address the IPP problem. However, it requires significant computational resources to simulate a diverse set of data and train the RL agent.

3.4 SGP-based IPP

Our SGP based sensor placement approach [74] has two key properties that are relevant to addressing the informative path planning (IPP) problem. First, the SGP approach can generate solution sensor placements for both discrete and continuous environments. Second, the approach is able to obtain sensor placement solutions on par with the ones obtained by maximizing mutual information (MI) but with significantly reduced computational cost.

However, the SGP based sensor placement approach does not consider the order in which the sensing locations are visited. Indeed, in IPP, we need to specify the order in which the sensing locations are visited and potentially also consider other constraints on the path, such as distance budget and velocity limits.

In the following, we first detail our approach to address the visitation order issue of the SGP based sensor placement approach for single-robot IPP. Then we explain how to impose routing constraints, such as a distance budget and velocity limits. After this, we generalize our approach to handle multi-robot IPP, and then finally address continuous sensing along the paths and modeling non-point FoV sensors.

3.4.1 Single-Robot IPP

We address the visitation order issue in spatially correlated environments by leveraging a travelling salesperson problem (TSP) solver [75]. In the most fundamental version of the single-robot IPP problem, we need not consider any constraints on the travel distance. Therefore, we first obtain s sensor placement locations using the SGP approach and then generate a path $\mathcal{P}$ that visits all the solution sensing locations by (approximately) solving the TSP, modified to allow for arbitrary start and end nodes:

$$\begin{aligned} \mathbf{X}_m &= \text{Continuous-SGP}(\theta, \mathcal{V}, s) \\ \mathbf{X}_m &= \text{TSP}(\mathbf{X}_m)\,. \end{aligned} \tag{3.2}$$

In spatio-temporally correlated environments, we do not even have to solve the TSP. This is because the generated solution sensor placements have an inherent visitation order since they span both space and time. However, such an approach could generate solutions that cannot be traversed by real-world robots with restrictions on their dynamics. The approach can handle a distance constraint if allowed to drop a few locations from the selected sensing locations, but it would do that without accounting for the information lost by dropping sensing locations. Therefore, we must develop a more sophisticated approach to address real-world IPP problems that have constraints such as a distance budget and velocity limits.

We do this by leveraging the differentiability of the SGP's optimization objec-

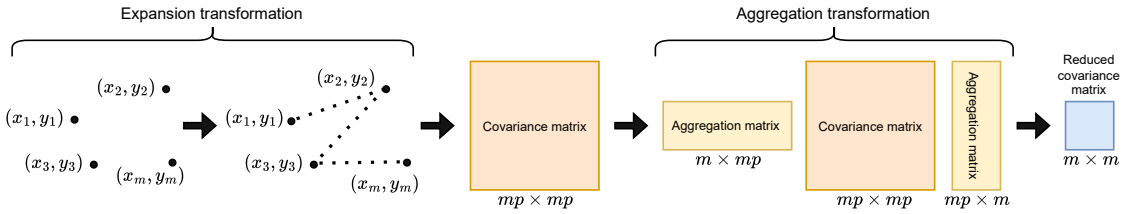


Figure 3.2: An illustration of the expansion and aggregation transformations used in IPP for continuous sensing robots.

tive $\mathcal{F}$ (Equation 1.9) with respect to the inducing points $\mathbf{X}_m$. The inducing points of the SGP $\mathbf{X}_m$, which we consider as the sensing locations, are used to compute the covariance matrix $\mathbf{K}_{mm}$, which is in turn used to compute the Nyström approximation matrix $\mathbf{Q}_{nn}$ in the objective function $\mathcal{F}$. We can impose constraints on the sensing locations by adding differentiable penalty terms dependent on the inducing points $\mathbf{X}_m$ to the objective function $\mathcal{F}$. Such an objective would still be differentiable and can be optimized using gradient descent.

We use the above method to impose constraints on even the solution *paths*. We do this by first solving the TSP on the SGP's initial inducing points $\mathbf{X}_m$ and treating them as an ordered set, which gives us an initial path that sequentially visits the inducing points. We then augment the SGP's objective function $\mathcal{F}$ with differentiable penalty terms that operate on the ordered inducing points for each path constraint and optimize the SGP to get the solution path. For instance, we can formulate the distance budget constraint as follows:

$$\begin{gathered}\hat{\mathcal{F}} = \mathcal{F} - \alpha \mathrm{ReLu}(\mathrm{PathLength}(\mathbf{X}_m) - c)\,, \\ \text{where } \mathrm{ReLu}(x) = \max(x, 0)\,.\end{gathered} \tag{3.3}$$

Here, PathLength is a function to obtain the total travel distance of the path that sequentially visits each of the inducing points $\mathbf{X}_m$ (treated as an ordered set) and α is a weight term used to scale the distance constraint penalty term. The ReLu function ensures that $\mathcal{F}$ remains unchanged if the path length is within the distance budget c and penalizes it only if the length exceeds the distance budget.

Similarly, we can accommodate additional constraints on the route, such as limits on the velocity and acceleration. In addition, we can trivially set predefined start and end points for the paths by freezing the gradient updates to the first and last inducing points of the SGP.

3.4.2 Multi-Robot IPP

We now address the multi-robot IPP problem. This is achieved by increasing the number of inducing points in the SGP. If we have r robots and need paths with s sampling locations each, we sample rs random points from the environment and find r paths by solving the vehicle routing problem (VRP, [76]). After resampling the paths so that each path has s points, this yields an ordered set with rs sensing locations that form the r initial paths. We initialize the SGP with these rs points as the inducing points. We then add the path constraints that operate on each path to the objective function $\mathcal{F}$ and optimize the SGP to get the set of r solution paths $\mathcal{P}$. The approach is shown in Algorithm 4. For spatio-temporal environments, we can also decouple the spatial and temporal inducing points to further reduce the computational cost of our approach and ensure that it has optimal waypoint assignments (see the Appendix [77]).

Algorithm 4: SGP-IPP: Multi-Robot IPP approach. μ is the SGP's mean, γ is the SGP learning rate, and VRP is the vehicle routing problem solver.

Input: Hyperparameters θ, domain of the environment $\mathcal{V}$, number of waypoints s, number of robots r, path constraints $\mathbf{C}$
Output: $\mathcal{P} = \{\mathcal{P}_i | \mathcal{P}_i \in \mathcal{V}, |\mathcal{P}_i| = s, i = 1, ..., r\}$

$\mathbf{X} \sim \mathcal{U}(\mathcal{V})$ // Uniformly draw unlabeled locations
$\mathbf{X}_m = \text{RandomSubset}(\mathbf{X}, rs)$ // Initialize $\mathbf{X}_m$
$\mathbf{X}_m = \text{VRP}(\mathbf{X}_m)$ // Get set of initial paths $\mathcal{P}$
// Add constraints to the SGP's optimization function $\mathcal{F}$
$\hat{\mathcal{F}} = \mathcal{F} - \alpha(\text{Constraints}(\mathbf{X}_m) - \mathbf{C})$
// Initialize SGP with the ordered inducing points
$\varphi = \mathcal{SGP}(\mu = 0, \theta, \mathbf{X}, \mathbf{y} = \mathbf{0}, \mathbf{X}_m, \hat{\mathcal{F}})$
Loop until *convergence* : $\mathbf{X}_m \leftarrow \mathbf{X}_m + \gamma \nabla \varphi(\mathbf{X}_m)$
return $\mathbf{X}_m$

3.4.3 IPP for Continuous and Non-point FoV Sensing Robots

Our approach so far has only considered discrete sensing robots with point sensors. However, there are instances where we require robots to continuously sense along

their paths, or to utilize sensors such as cameras with non-point fields of view (FoV). While continuous and non-point FoV sensing robots can use paths optimized for discrete sensing robots with point sensors, explicitly optimizing paths for continuous and non-point FoV sensing robots is likely to yield more informative paths.

A naive approach to modeling continuous sensing robots is to use a large number of inducing points. However, such an approach would limit scalability due to the cubic complexity of SGPs with respect to the number of inducing points. Additionally, it cannot handle non-point FoV sensors, as even if the additional inducing points are initialized to match the sensor's FoV, they would not retain the FoV shape after optimization. A key advantage of generalizing the SGP-based sensor placement approach [74] to IPP is that we can leverage the properties of GPs and SGPs [2, 78, 7, 8]. We describe two such properties and how they can be used to address IPP for continuous and non-point FoV sensing robots.

First, we utilize the property that the inducing points of SGPs can be transformed with any non-linear function and still be optimized using gradient descent. We can employ such transformations to approximate the information along solution paths. To achieve this, we parameterize the m inducing points of the SGP as the sensing locations for a discrete point sensing robot's path. Then, we apply a transformation—the expansion transformation T_{exp}—to interpolate p additional points between every consecutive pair of inducing points that form the robot's path $\mathbf{X}_{mp} = T_{\text{exp}}(\mathbf{X}_m)$, resulting in mp inducing points (actually $(m-1)p$ points, denoted as mp to simplify notation). We then utilize the mp inducing points $\mathbf{X}_{mp}$ to compute the SGP's objective function $\hat{\mathcal{F}}$ with path constraints. Note that the interpolation operation is differentiable, enabling us to compute gradients for the m inducing points $\mathbf{X}_m$. This approach enables us to consider the information gathered along the entire path.

Next, we leverage the property of GPs that they are closed under linear transformations [2, 58]. We use the aggregation transformation T_{agg}, which aggregates

(via averaging) the SGP's covariances corresponding to the p inducing points that approximate the information between every consecutive pair of waypoints of the path (i.e., the m inducing points), thereby reducing the size of the covariance matrix from $mp \times mp$ back to $m \times m$. Specifically, we first employ the expansion transformation T_{exp} on the m inducing points to map them to a larger set of mp points. Then, we apply the aggregation transformation T_{agg} to the covariance matrices built using the mp points. These covariances are utilized to compute $\mathbf{Q}_{nn}$, which in turn is used to compute the SGP's objective function (Equation 1.9):

$$\mathbf{Q}_{nn} = \mathbf{K}_{n\times mp} T_{\text{agg}} (T_{\text{agg}}^{\top} \mathbf{K}_{mp\times mp} T_{\text{agg}})^{-1} T_{\text{agg}}^{\top} \mathbf{K}_{mp\times n} \,. \tag{3.4}$$

Here, $\mathbf{K}_{n\times mp}$ represents the covariance matrix between the n training set inputs and the mp inducing points. The aggregation transformation reduces the size of the covariance matrix $\mathbf{K}_{mp\times mp}$ before inversion. Consequently, the matrix inversion cost is reduced to $\mathcal{O}(m^3)$ from $\mathcal{O}(m^3p^3)$, allowing us to benefit from both the expansion transformation, which enables modeling of continuous sensing robots, and the reduced computational cost from the aggregation transformation. Note that the information along the path between the waypoints is retained in the aggregated covariances. Additionally, we observed that the aggregation transformation stabilized the gradients during the optimization of the inducing points. The approach is illustrated in Figure 3.2 and shown in Algorithm 5.

We can use the transformations detailed above to handle non-point field of view (FoV) sensing robots as well. This is accomplished by employing the expansion transformation to map each of the m inducing points to p points that approximate the sensor's FoV area. Additionally, since the gradients are propagated back to the original m inducing points $\mathbf{X}_m$, the method retains the FoV shapes. Furthermore, we can leverage this property to model sensors with integrated observations such as gas sensors [17, 79], where the labels are modeled as $y_i = ||\mathbf{w}_i|| \int_0^1 f(\mathbf{w}_i t + \mathbf{z}_i) dt + \epsilon_i$,

Algorithm 5: Expansion and aggregation transformation based approach for obtaining non-point FoV sensor placements. Here θ are the hyperparameters learnt from either historical data or expert knowledge, Φ is a random distribution defined over the bounds of the environment $\mathcal{V}$, s is the number of required sensors, n is the number of random locations used to train the SGP, and γ is the SGP learning rate. T_{exp} and T_{agg} are the expansion and aggregation transformations, respectively.

Input: Hyperparameters θ, domain of the environment $\mathcal{V}$, number of waypoints s, number of random unlabeled samples n used to train the SGP, expansion transformation T_{exp}, and aggregation transformation T_{agg}

Output: Solution sensor placements $\mathcal{A} \subset \mathcal{V}$, where $|\mathcal{A}| = s$

$\mathbf{X} = \{\emptyset\}$; `// Initialize empty set to store SGP training set`

repeat

  `// Draw` n `random unlabeled locations from the environment`

  $\mathbf{x} \sim \mathcal{U}(\mathcal{V})$

  $\mathbf{X} \leftarrow \mathbf{X} \cup \{\mathbf{x}\}$

until $|\mathbf{X}| = n$;

$\mathcal{D} = (\mathbf{X}, \mathbf{y} = \mathbf{0})$;

`// Generate SGP training dataset with` 0 `labels`

$\mathbf{X}_m = \text{RandomSubset}(\mathbf{X}, s)$;

`// Initialize` s `inducing points at random locations`

$\mathbf{X}_m \leftarrow \text{RandomTheta}(\mathbf{X}_m, s)$;

`// Add random sampled angles as the rotation parameter of each inducing point`

$\varphi = \mathcal{SGP}(\mu = 0, \theta, \mathcal{D}, \mathbf{X}_m)$;

`// Initialize a SVGP` φ `with` 0 `mean, hyperparameters` θ`, training set` $\mathcal{D}$`, and inducing points` $\mathbf{X}_m$

repeat

  $\mathbf{X}_{mp} = T_{\text{exp}}(\mathbf{X}_m)$;

  `// Use the expansion transformation` T_{exp} `to map the` m `inducing points` $\mathbf{X}_m$ `in the point parametrization to` mp `points with FoV parametrization`

  $\mathbf{Q}_{nn} = (\mathbf{K}_{n \times mp} T_{\text{agg}})(T_{\text{agg}}^{\top} \mathbf{K}_{mp \times mp} T_{\text{agg}})^{-1}(T_{\text{agg}}^{\top} \mathbf{K}_{mp \times n})$;

  `// Use the aggregation transformation` T_{agg} `to reduce the covariances`

  $\mathbf{X}_m \leftarrow \mathbf{X}_m + \gamma \nabla \varphi(\mathbf{Q}_{nn})$;

  `// Optimize the point parametrized inducing points` $\mathbf{X}_m$ `by maximizing the SVGP's ELBO` $\mathcal{F}$ `using gradient methods with a learning rate of` γ`. We compute the ELBO using the` $\mathbf{Q}_{nn}$ `computed above`

until *convergence*;

return $\mathbf{X}_m$

with $\mathbf{z}_i$ as the start point of a line segment along which the data is integrated, and $\mathbf{w}_i$ giving the direction and length of the line segment. Similarly, we can efficiently model complex path parametrizations, such as using splines to obtain smooth paths, account for sensors such as cameras whose FoV varies with height from the ground, and even model FoVs that consider the shape of the surface, such as stereo vision cameras when used to scan 3D surfaces. Please refer to Appendix B for implementation details of the expansion transformation.

3.4.4 Space-Time Decomposition

When considering multiple robots in spatio-temporal environments, our approach can be further optimized to reduce its computation cost and can be show to have optimal waypoint assignments. When optimizing the paths with t waypoints for r robots, instead of using rt inducing points $\mathbf{X}_m \in \mathbb{R}^{r\times t\times(d+1)}$, we can decouple the spatial and temporal inducing points into two sets—spatial and temporal inducing points. The spatial inducing points $\mathbf{X}_{\text{space}} \in \mathbb{R}^{rt\times d}$ are defined only in the d-spatial dimensions. The temporal inducing points $\mathbf{X}_{\text{time}} \in \mathbb{R}^{t}$ are defined across the time dimension separately. Also, we assign only one temporal inducing point for the r robots at each time step. This would constrain the r paths to have temporally synchronised waypoints. We then combine the spatial and temporal inducing points by mapping each temporal inducing point to the corresponding r spatial inducing points, forming the spatio-temporal inducing points $\mathbf{X}_m \in \mathbb{R}^{r\times t\times(d+1)}$.

The approach allows us to optimize the inducing points across space at each timestep and the timesteps separately. It ensures that there are exactly r inducing points at each time step and also reduces the number of variables that need to be optimized. Specifically, we only have to consider t temporal inducing points, rather than rt inducing points along the temporal dimension. During training, we can use backpropagation to calculate the gradients for the decomposed spatial and temporal inducing points through the spatio-temporal inducing points.

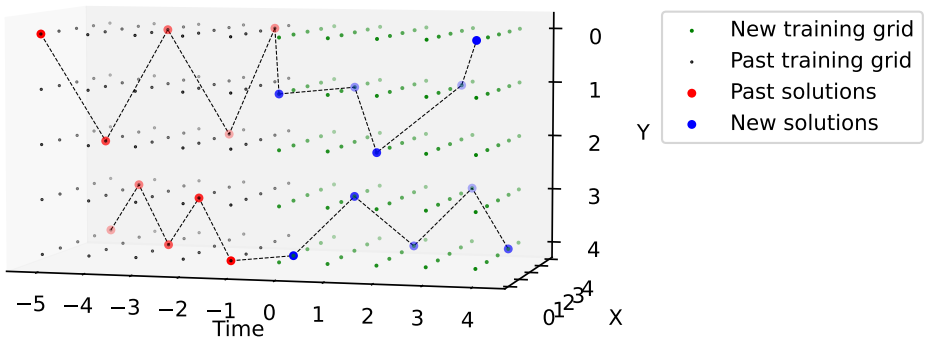


Figure 3.3: A schematic illustration of the IPP approach for multiple robots (2 robots). Note that the past training grid is only for visual interpretability and is not used while training the SGP to get the new path.

An added advantage of this decomposition is that we can leverage it to prove that the solution paths have optimal waypoint transitions. We do this by setting up an assignment problem that maps the r waypoints at timestep i to the r waypoints at timestep $i+1$. We calculate the assignment costs using pairwise Euclidean distances and get the optimal waypoint transitions for each of the r paths by solving for the assignments [44]. We repeat this procedure for all t timesteps. If the transitions are optimal, the approach would return the original paths, and if not, we would get the optimal solution. The pseudocode of the approach is shown in Algorithm 6, and Figure 3.3 illustrates our approach.

Algorithm 6: Assignment problem based approach to get optimal waypoint transitions for r paths.

Input: Inducing points $\mathbf{X} \in \mathbb{R}^{r \times t \times (d+1)}$
Output: Waypoints of paths $\mathbf{X} \in \mathbb{R}^{r \times t \times (d+1)}$

for $i \leftarrow 0$ **to** $t-1$ **do**
  $\mathbf{C} = \mathbf{0}^{r \times r}$
  for $j \leftarrow 1$ **to** r **do**
    for $k \leftarrow 1$ **to** r **do**
      $\mathbf{C}[j][k] \leftarrow ||\mathbf{X}[j, i, :d] - \mathbf{X}[k, i+1, :d]||_2$
  $A = \mathcal{H}(\mathbf{C})$ // Solve the assignment problem [44]
  // Re-index the inducing points at time $i+1$
  $\mathbf{X}[:, i+1, :d+1] \leftarrow \mathbf{X}[A, i+1, :d+1]$
return X

3.4.5 IPP with Past Data

In a real-world scenario, it is possible that a robot has collected data from a sub-region of the environment. In such cases, in addition to updating our kernel parameters, it would be beneficial to explicitly incorporate the data into our path planning approach. We do this by adding more inducing points—auxiliary inducing points—to the SGP in addition to the m inducing points used to get a path with m sampling locations. The auxiliary inducing points are initialized at the locations of the past data samples, and their temporal dimension is set to negative numbers indicating the time that has passed since the corresponding data sample was collected. During training, the auxiliary inducing points are not optimized; only the original m inducing points used to form the path are optimized. Therefore, the approach accounts for past data, including when it was collected, as the points collected further in the past would be less correlated with the SGP's training data consisting of random samples restricted to the positive timeline. Note that this approach is also suited to address online variants of IPP.

3.5 Experiments

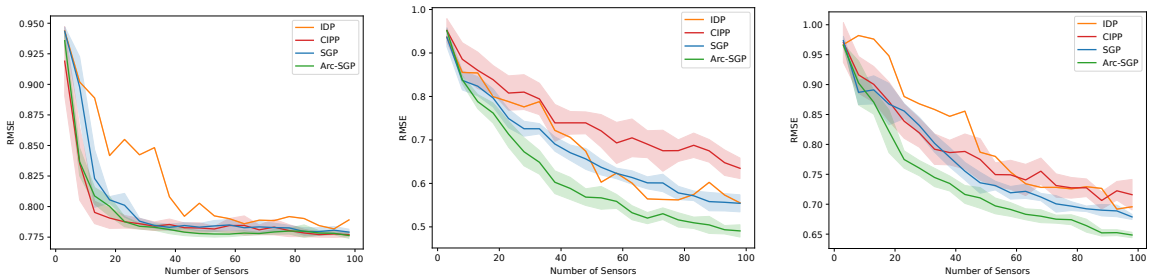
We first demonstrate our approach for the unconstrained single robot IPP problem on the ROMS ocean salinity [48] and US soil moisture [47] datasets. The ROMS dataset contains salinity data from the Southern California Bight region, and the US soil moisture dataset contains moisture readings from the continental USA. All experiments were benchmarked on a workstation with an 18-core Intel(R) Xeon(R) Gold 6154 3.00GHz CPU and 128 GB of RAM.

3.5.1 Single-Robot IPP

We benchmarked our SGP based IPP approach (SGP) that optimizes a path for discretely sensing s locations and our transformation based generalization of the SGP based IPP approach (Arc-SGP) that optimizes the paths while accounting for the

information collected along the entire path. We also benchmarked two baseline approaches—the Information-Driven Planner (IDP, Ma et al. [56]) and Continuous-Space Informative Path Planner (CIPP, Hitz et al. [36]). IDP leverages discrete optimization to iteratively find discrete sensing locations that maximize mutual information (MI), and CIPP leverages CMA-ES, a genetic algorithm, to find informative sensing locations that maximize MI in continuous spaces.

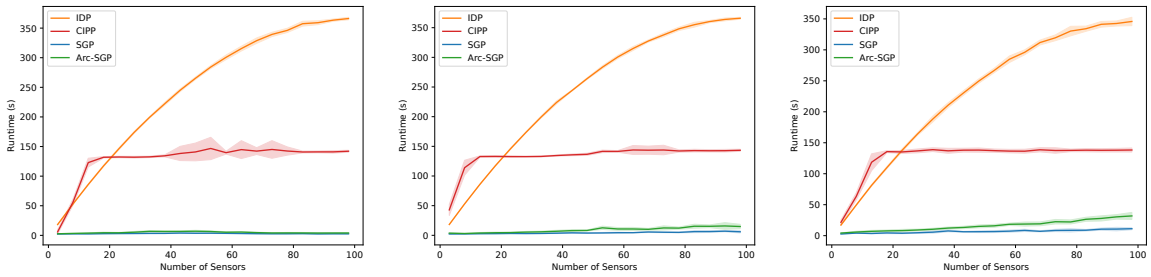
An RBF kernel [2] was used to model the spatial correlations of the datasets (the baselines use it to measure MI). We evaluated the paths by gathering the ground truth data along the entire generated solution paths (i.e., by continuous sensing robots) and estimating the state of the whole environment from the collected data. The root-mean-squared error (RMSE) between the ground truth data and our estimates was used to quantify the quality of solution paths. We generated solution paths for both the datasets with the number of path sensing locations ranging from 3 to 100 in increments of 5. The experiment was repeated 10 times. The mean and standard deviation of the RMSE and runtime results on the ROMS and US soil moisture datasets are shown in Figure 3.4 and Figure 3.5, respectively.



(a) Precipitation (b) ROMS (c) Soil

Figure 3.4: RMSE results for single robot IPP with the IDP, CIPP, SGP, and Arc-SGP approaches on the precipitation, ROMS and US soil datasets.

As we can see, our SGP approach is consistently on par or better than the two baselines in terms of RMSE, and our Arc-SGP approach has a considerably lower RMSE than the other approaches in all cases. Also, both our approaches substan-

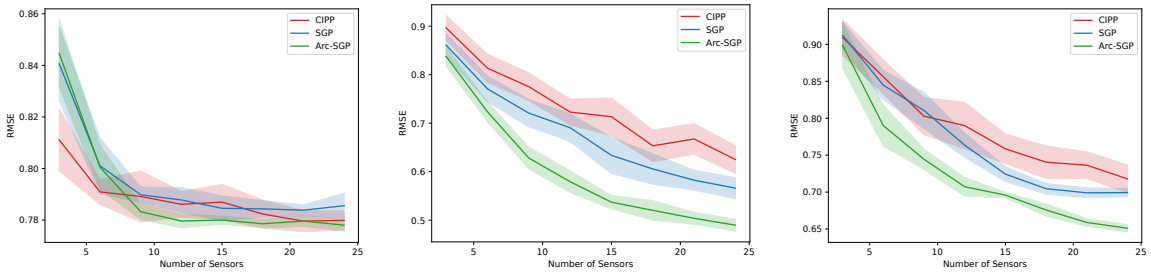


(a) Precipitation (b) ROMS (c) Soil

Figure 3.5: Runtime results for single robot IPP with the IDP, CIPP, SGP, and Arc-SGP approaches on the precipitation, ROMS and US soil datasets.

tially outperform the baselines in computation time (up to 35 times faster). In both the baselines, a significant amount of computation time is spent on computing MI, while our SGP approach's objective approximates the same in a computationally efficient manner (detailed in our foundational work [74]). Indeed, the MI computation cost is the key reason why both IDP and CIPP cannot scale to spatio-temporally correlated environments, since even with a coarse discretization, it would be far too computationally expensive. Also, since our approaches rely on gradient information, they are significantly faster to converge compared to the discrete and genetic algorithm based baseline approaches.

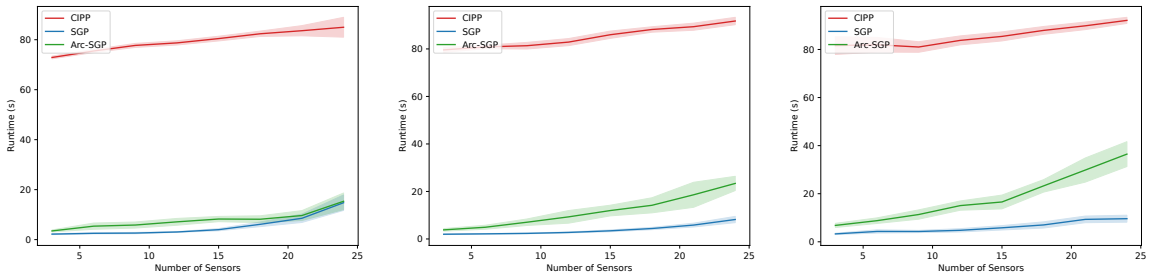
3.5.2 Multi-Robot IPP



(a) Precipitation (b) ROMS (c) Soil

Figure 3.6: RMSE results for four robot IPP with the CIPP, SGP, and Arc-SGP approaches on the precipitation, ROMS and US soil datasets.

We now demonstrate our approach for multi-robot IPP. We used the same kernel



(a) Precipitation (b) ROMS (c) Soil

Figure 3.7: Runtime results for four robot IPP with the CIPP, SGP, and Arc-SGP approaches on the precipitation, ROMS and US soil datasets.

parameters as we did in the previous experiments. The solution paths were generated for four robots with the number of optimization waypoints ranging from 3 to 25 in increments of 5 for each robot's path. We evaluated the SGP, Arc-SGP, and CIPP methods, which support multi-robot IPP. The RMSE and runtime results are shown in Figure 3.6 and Figure 3.7, respectively. Our SGP approach consistently performs on par with or better than the CIPP approach in terms of RMSE. Additionally, Arc-SGP achieves notably lower RMSE compared to both SGP and CIPP, as it explicitly optimizes the paths for continuous sensing. Moreover, both our approaches—SGP and Arc-SGP—significantly outperform CIPP (up to 26 times faster) in terms of compute time.

3.5.3 IPP with Distance Constraints

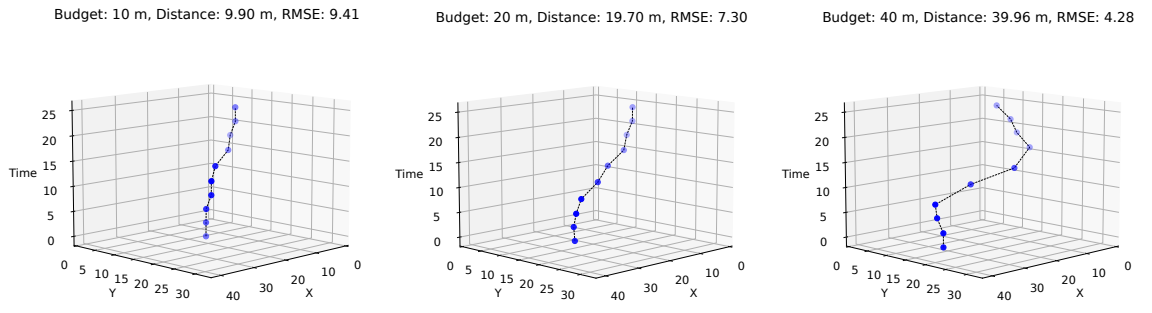


Figure 3.8: Data collection paths generated using a spatio-temporal kernel function for different distance budgets.

We next show our approach for spatio-temporal IPP with a distance constraint. A Gaussian process was used to sample dense spatio-temporal temperature data. We used an RBF kernel with a length scale of 7.70 m, 19.46 m, and 50.63 mins along the x, y, and temporal dimensions, respectively. We generated paths by optimizing the inducing points in our SGP approach with distance budgets of 10 m, 20 m, and 40 m; the results are shown in Figure 3.8. Our approach consistently saturates the distance budget without exceeding it to get the maximum amount of new data, evident from the paths' RMSE scores. We also show the paths generated for three robots in the same environment (Figure 3.9). We do not show the reconstructions since the data is spatio-temporal, which is difficult to show in 2D.

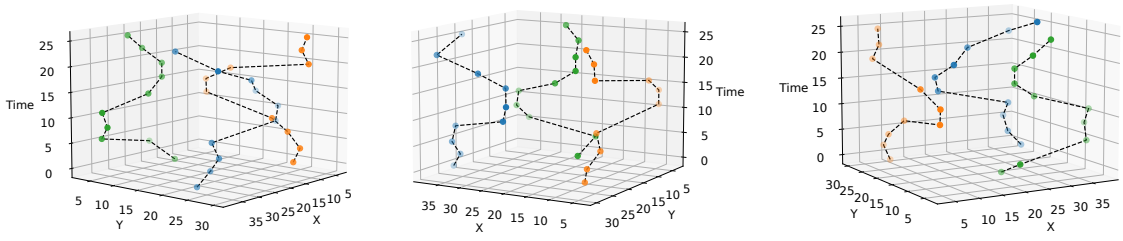


Figure 3.9: Three different views of our multi-robot IPP solution paths, with path lengths of 47.29 m, 47.44 m, and 47.20 m. The data from all 3 paths gave us an RMSE of 0.34.

3.5.4 IPP with a Non-Point Height-dependent FoV

Figure 3.10 shows our SGP approach for a discrete sensing robot, i.e., it senses only at the path's vertices (blue points). We considered a 3D environment with densely sampled elevation data and parametrized the path so that we account for the robot's sensing FoV area to be a function of the robot's height from the ground. We used an RBF kernel with a length scale of 3 m (details in the Appendix [77]).

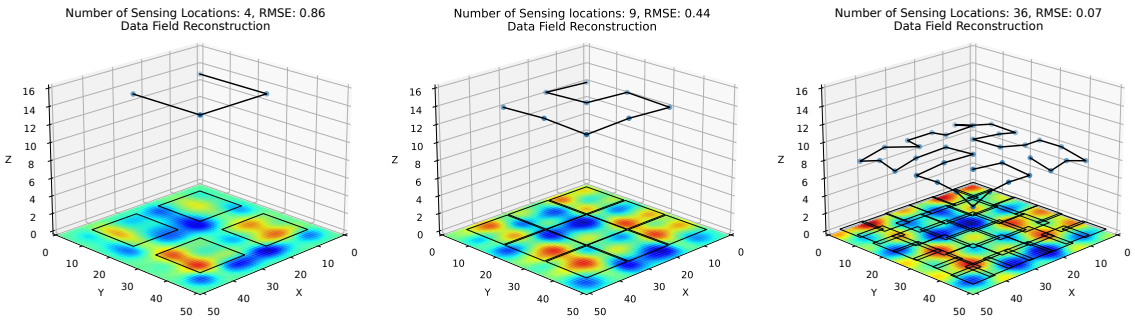


Figure 3.10: Solution paths for a discrete sensing robot with a square height-dependent FoV area (black squares) sensor. The solution paths adjust the sensor height to ensure a good balance between the ground sampling resolution and the coverage area.

3.5.5 IPP with Past Data

We demonstrate the approach with past data. We used 5 data samples as the past data and generated new informative paths with the distance budgets set to 10, 20, and 40. The results are shown in Figure 3.11; the red points are the past data samples. As we can see, our new solution paths are shifted to avoid collecting data at the location of the past data. Therefore, our solution paths have lower RMSE scores than those generated without past data information.

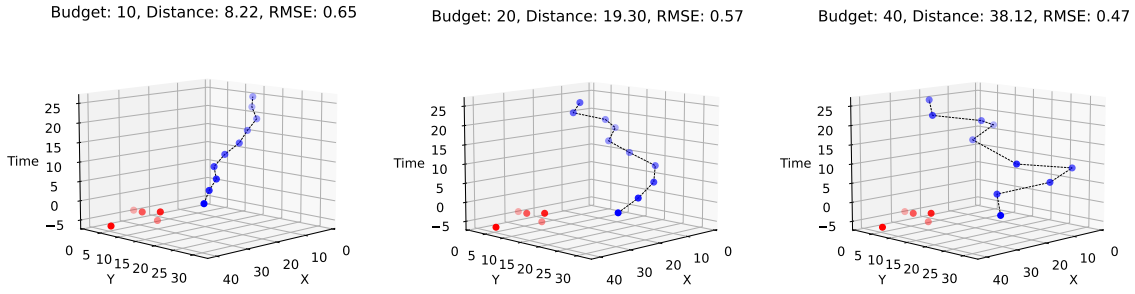


Figure 3.11: Five points were used as past data (red points). Data collection paths generated using a spatio-temporal kernel function for different distance budgets.

3.5.6 Sensor Placement for Integrated Non-Point FoV Sensors

Finally, as we mentioned in Section 3.4.3, our method expansion and aggregation transformations can be used to model sensors with non-point FoV and integrated

observations as well. We demonstrate this on the sensor placement for the sparse view computed tomography (CT) scanning problem [79]. A third-generation fan beam CT scanner [80] projects X-ray beams that fan out at a fixed angle (Figure 6.2). On the opposite side of the projector, a sensor array captures data integrated along each beam and encoded by the received X-ray intensity. The projector and sensor array pair rotate around a circular area of interest, and the collected data is then used to reconstruct the underlying spatially resolved data. Being able to compute informative sensor placements efficiently is especially useful when the region of interest is a sub-region of the observation space (e.g., if we only need to scan a specific organ), as we will be able to optimize the sensor placements in real-time, thereby reducing the amount of harmful X-ray exposure to patients while also retaining reconstruction quality.

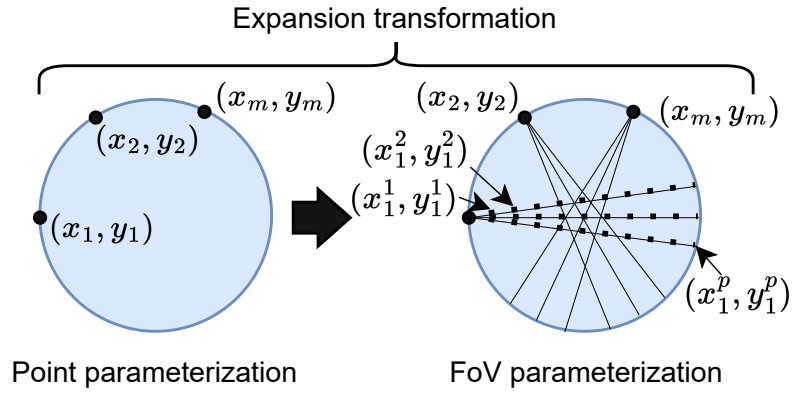
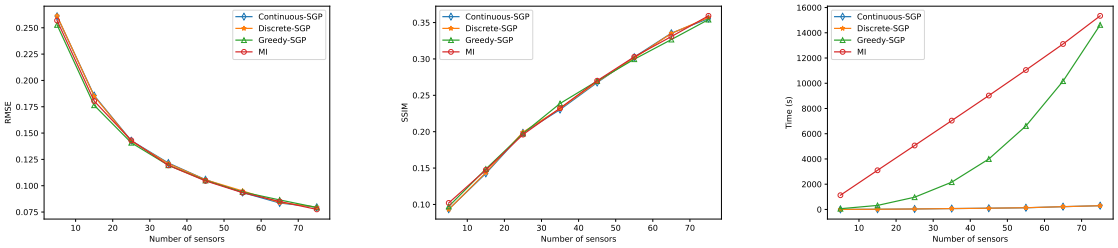


Figure 3.12: An illustration of the expansion transformation used for sensor placement in sparse view CT-scanning.

We used the COVID-19 CT scan dataset which contains lung scans from 10 patients, with each containing 301 slices [81] to benchmark our sparse view CT scan sensor placements. We used a fan beam CT projection [80] with 750 detectors with a width of 2. The source projection and the detector distance were set to 400. We used filter back projection [80] from the ASTRA Toolbox [82] to generate our reconstructions using the data from only the placement locations. The kernel parameters



(a) RMSE (b) SSIM (c) Runtime

Figure 3.13: RMSE, SSIM, and runtime results for the CT scan dataset.

were learned from phantom CT images.

We used the whole observation space as the region of interest. Figure 3.12 illustrates the expansion transformation, and we used a mean operation as the aggregation transformation. We computed the solution sensor placements for 5 to 75 sensors (in increments of 10) using our approaches (with p set to 20) and the mutual information-based approach presented in [79] as a baseline. [79] addressed sensor placement for sparse view CT-scanning using the mutual information-based approach by [13]. They leveraged the closed-form nature of Gaussian processes under linear transformations to extend the MI approach to handle the data-integrating sensor model of CT scanners. However, the method still relies on discrete optimization, which makes the approach computationally expensive and ill-suited for real-time applications. We present the RMSE and SSIM scores of the CT scan dataset reconstructions obtained using the solution sensor placement locations, and the sensor placement algorithm runtimes in Figure 3.13, respectively.

Our results show that our approaches are significantly faster (up to 53 times) while maintaining the reconstruction quality obtained using the computationally intensive MI-based approach. In the Greedy-SGP case, the method is still faster than the MI approach despite using a greedy optimization method. Also, the Continuous-SGP and Discrete-SGP approaches take a fraction of the computation time of the baseline MI approach and take approximately constant time with increasing number of sensing

locations.

3.6 Conclusion

We presented an efficient continuous space approach to informative path planning using sparse Gaussian processes that can address various challenges related to monitoring in spatially and spatio-temporally correlated environments. Our approach can model routing constraints, and handle discrete and continuous sensing robots with arbitrary FoV shapes. Furthermore, our method generalizes to multi-robot IPP problems as well. We demonstrated that our approach is fast and accurate for IPP on real-world data. We also presented our IPP solutions for different distance budgets, multi-robot scenarios, and with non-point FoV sensing robots. Our future work will build upon this approach to extend its applicability to online and decentralized IPP problems.

CHAPTER 4: Online and Decentralized Heterogeneous Multi-Robot Informative Path Planning in Continuous and Discrete Environments for Data Field Estimation

4.1 Introduction

Informative Path Planning (IPP) is a fundamental problem in robotics. The problem requires finding paths to obtain the maximal amount of novel data about the underlying data field of interest. One must also ensure that all path constraints, such as distance-budget limits and boundary constraints, are satisfied. Moreover, it is often the case that the environment is unexplored and there is no training data available. In such cases, online IPP variants, which can plan initial paths and update the paths by learning from the data being collected along the paths, need to be developed. Also, in persistent monitoring problems, where robots must continuously monitor the environment for tasks such as pollution tracking in lakes and rivers, online approaches that can detect and adapt to changes in the data field mid-mission are crucial.

The IPP problem is particularly relevant for environmental monitoring, where one must monitor phenomena like ozone concentration, soil moisture, and ocean salinity in large environments [57, 56, 15, 17, 74, 77]. It is also prevalent in surface inspection tasks [19], where robots are used to inspect 3D structures like bridges, dams, aircraft wings, and pipeline insulation. IPP can also be used for collecting aerial images that result in accurate semantic mapping [59].

Given the significance of the IPP problem, multiple authors have addressed it [36, 56, 68]. Most approaches compute the informativeness of locations using mutual information (MI) calculated with Gaussian processes (GPs). But, MI is not differentiable with respect to the sensing locations on the path(s). Therefore, computationally expensive discrete optimization methods such as greedy algorithms, Bayesian opti-

mization, and genetic algorithms are employed to optimize the solutions.

A key limitation of the above GP-based approaches is their inability to scale to large environments due to high computational costs. Furthermore, computing MI using GPs requires a priori knowledge of the GP hyperparameters that accurately model the data field in the environment. Online IPP approaches typically address this issue by learning the relevant hyperparameters from data collected along the path using another GP or sparse GP, followed by updating the remaining waypoints with discrete optimization methods [36, 56]. However, this simply propagates the offline approach's compute cost limitation to the online setting, further exacerbated by the added cost of hyperparameter learning, thereby diminishing their online performance. Further, only a few prior approaches addressed decentralized multi-robot IPP, which is crucial in areas with poor communication networks [83, 84]. Finally, IPP with heterogeneous robot teams is not well addressed, which is particularly important if the robots have different distance budgets and sensor models.

We present a computationally efficient online and decentralized heterogeneous multi-robot IPP approach, enabling the deployment of robot teams in a fully decentralized manner. This is achieved with low individual robot computation and communication requirements, while also accommodating heterogeneous robots with different path constraints and sensor models. Our approach achieves this by leveraging sparse Gaussian processes (SGPs) and streaming sparse Gaussian processes (SSGPs). Our method can even find solution paths for robots with continuous sensing and non-point field of view sensors, as well as handle dynamic environments, including spatio-temporal environments.

4.2 Online and Decentralized Heterogeneous Multi-Robot Informative Path Planning Problem

We address online IPP to monitor an environment $\mathcal{V} \subseteq \mathbb{R}^d$ representing a phenomenon such as temperature. We have r robots and must find the set $\mathcal{P}$ with r

paths, one for each robot, so that the data $\mathbf{y} \in \mathbb{R}$ collected at these locations is sufficient to accurately estimate the phenomenon at every location in the environment. We use the root-mean-square error (RMSE) of the estimates as the measure of accuracy. Since we cannot directly minimize the RMSE, we formulate this problem as one where we want to find the paths $\mathcal{P}$ that maximize the amount of information I. Here I is any function that is a good proxy for accuracy and can be computed without the ground truth labels. Moreover, we also consider constraints $\mathbf{C}$ such as distance budget limits and boundary constraints on the paths:

$$\begin{aligned} \mathcal{P}^* = \underset{\{\mathcal{P}_i \in \psi, i=1,\dots,r\}}{\arg\max} \quad & I(\cup_{i=1}^{r} \text{SAMPLE}(\mathcal{P}_i)), \\ \text{s.t. } & \text{Constraints}(\mathcal{P}_{i=1,\dots,r}) \leq \mathbf{C} \end{aligned} \tag{4.1}$$

Here ψ is the space of paths contained within the environment $\mathcal{V}$. The SAMPLE function returns the sensing points at the robot path waypoints when modeling a discrete sensing robot, and it returns all the points along the path when modeling a continuous sensing robot. In addition, we also consider point sensors such as temperature probes, and non-point sensors that can have any field-of-view (FoV) shape, such as a thermal vision camera with a rectangular FoV. Moreover, we assume that no training data is available from the target environment and that there is no robot-to-robot communication capability. Finally, we also consider IPP for heterogeneous robot teams, where each robot can have a different sensing model and path constraints. This online IPP problem generalizes the offline IPP problem addressed in our previous work [77] by requiring updates to the solution path in response to newly gathered online data, and necessitating a decentralized multi-robot approach that can accommodate heterogeneous robot sensing models and path constraints.

4.3 Related Work

The Informative Path Planning (IPP) problem is known to be NP-hard [62]. Therefore, only suboptimal solutions can be found for most real-world problems. Many IPP approaches use mutual information (MI), an information metric that is submodular [63, 61, 57], as the optimization objective. MI can be computed using GPs with known kernel parameters. But the approach is computationally expensive ($\mathcal{O}(|D|^3)$, where D is the discretized environment).

Singh et al. [65] proposed a recursive-greedy algorithm that maximized MI for single and multi-robot IPP. Bottarelli et al. [66] developed active learning-based IPP algorithms with a complexity of $\mathcal{O}(|D|^5)$. Hollinger and Sukhatme [60] presented offline IPP algorithms for continuous spaces that maximized MI using rapidly-exploring random trees (RRT) and derived asymptotically optimal guarantees. Miller et al. [67] addressed continuous-space offline IPP with known utility functions using an ergodic control algorithm. Hitz et al. [36] developed an online multi-robot IPP approach based on a genetic algorithm that could optimize the sensing locations in continuous spaces given a utility function. Francis et al. [37] and Vivaldini et al. [69] leveraged Bayesian optimization for single robot IPP in continuous spaces. However, similar to discrete optimization and genetic algorithm based approaches, the method was computationally expensive and limited the approach's scalability. Ma et al. [56] solved the IPP problem by maximizing MI using dynamic programming and used an online variant of sparse Gaussian processes for learning the model parameters.

Schmid et al. [85] addressed online IPP for 3D reconstruction. They utilized a quadratic function of the distance from the camera to model the uncertainty in the environment and optimized it using RRTs in an online algorithm. Zhu et al. [19] also addressed IPP for 3D reconstruction and used a probabilistic estimate of variance as the uncertainty measure. Mishra et al. [70] and Berget et al. [86] addressed online IPP by selecting locations with high variance as the solution waypoints. Moon et al. [87]

used a sampling-based method to reduce entropy. However, variance and entropy-based approaches do not result in good sensing locations when compared to MI-based approaches [13]. Ghassemi et al. [83] and Newaz et al. [84] addressed decentralized IPP using Bayesian optimization with an acquisition function that selects locations estimated to have a large function value. Such approaches are suited for tracking modes of data fields but might fall short when considering full environment monitoring problems. Cao et al. [88] and Rückin et al. [73] leveraged deep reinforcement learning (DRL) to address the online IPP problem. However, they require simulating a diverse set of data and utilize significant computational resources to train the RL agent on the data before deployment.

4.4 SGP-based Online and Decentralized Heterogeneous Multi-Robot IPP

We can generalize our SGP-based offline IPP approach to perform online IPP by iteratively alternating between optimizing the solution path(s) with the offline planner and learning the SGP's hyperparameters using another separate Gaussian process trained on the data collected along the traversed portion of the path(s). Since the SGP-based offline IPP approach is significantly faster than earlier discrete optimization-based approaches, our online approach will also be significantly faster than the earlier online counterparts.

However, a naive implementation of this approach has a few limitations. First, the computational cost of training the GP to learn the hyperparameters from the collected data would keep increasing until it becomes infeasible to compute after about 10,000 data samples. Since most sensors of interest, such as thermal cameras and ocean salinity sensors, operate well above 10Hz, this data limit would be reached very quickly when modeling continuous sensing robots, i.e., robots that sense along the whole path. Second, when performing multi-robot IPP, the method would be limited to centralized planning and require each robot to have a large-bandwidth communication link to the central planner to stream all its collected data.

4.4.1 Fast Online Hyperparameter Estimation

We address the hyperparameter estimation problem by leveraging streaming sparse Gaussian processes (SSGPs, [89]). SSGPs are a principled approach to handle streaming data, i.e., data that arrives sequentially in batches. With SSGPs, the hyperparameter updates can be computed using only the current batch of data $\mathbf{y}_{\text{new}}$ and the previously computed variational distribution over all the past data $\mathbf{y}_{\text{old}}$. Moreover, the method leverages a sparse approximation to the streaming data; therefore, it does not require access to the whole dataset and significantly reduces the training time.

At each update step, to approximate all the data seen until the current update step, the method computes the new optimal variational posterior distribution $q_{\text{new}}(\hat{\mathbf{f}})$, using the new batch of data $\mathbf{y}_{\text{new}}$, to replace the old variational distribution $q_{\text{old}}(\hat{\mathbf{f}})$. The optimal distribution can be computed by minimizing the following Kullback–Leibler divergence:

$$\text{KL}\left[q_{\text{new}}(\hat{\mathbf{f}})||p(\hat{\mathbf{f}}|\mathbf{y}_{\text{old}}, \mathbf{y}_{\text{new}})\right] \tag{4.2}$$

Here, $\hat{\mathbf{f}}$ are all the latent variables corresponding to the data. The above can be analytically minimized to obtain the following optimal posterior variational distribution:

$$\begin{aligned}
&q_{\text{opt}}(\mathbf{b}) = p(\mathbf{b})\mathcal{N}(\hat{\mathbf{y}}; \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}}\mathbf{K}_{\mathbf{bb}}^{-1}\mathbf{b}, \Sigma), \\
&\text{where,} \\
&q(\mathbf{a}) = \mathcal{N}(\mathbf{a}; \mathbf{m}_{\mathbf{a}}, \mathbf{S}_{\mathbf{a}}),\ p(\mathbf{y}_{\text{new}}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}_{\text{new}}; \mathbf{f}, \sigma^2 I\right), \\
&\hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_{\text{new}} \\ \mathbf{D}_{\mathbf{a}}\mathbf{S}_{\mathbf{a}}^{-1}\mathbf{m}_{\mathbf{a}} \end{bmatrix}, \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} = \begin{bmatrix} \mathbf{K}_{\mathbf{fb}} \\ \mathbf{K}_{\mathbf{ab}} \end{bmatrix}, \\
&\Sigma = \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \mathbf{D}_{\mathbf{a}} \end{bmatrix},\ \text{and } \mathbf{D}_{\mathbf{a}} = \left[\mathbf{S}_{\mathbf{a}}^{-1} - \mathbf{K}_{\mathbf{aa}}^{-1}\right].
\end{aligned} \tag{4.3}$$

Here, $\mathbf{a}$ and $\mathbf{b}$ represent the latent variables corresponding to the old inducing points (i.e., from the previous update step) and new inducing points of the posterior variational distributions, respectively. The $\mathbf{f}$ are the latents corresponding to $\mathbf{y}_{\text{new}}$. The solution, the optimal posterior $q_{\text{opt}}(\mathbf{b})$, has a prior term $p(\mathbf{b})$ to regularize the posterior and a likelihood term $\mathcal{N}(\hat{\mathbf{y}})$, which predicts the data labels using the current inducing variables $\mathbf{b}$ and the old variational distribution $q(\mathbf{a})$. This ensures that the information from the old variational distribution $q(\mathbf{a})$ is retained while also capturing the new information in $\mathbf{y}_{\text{new}}$. Please refer to [89] for details of the derivation.

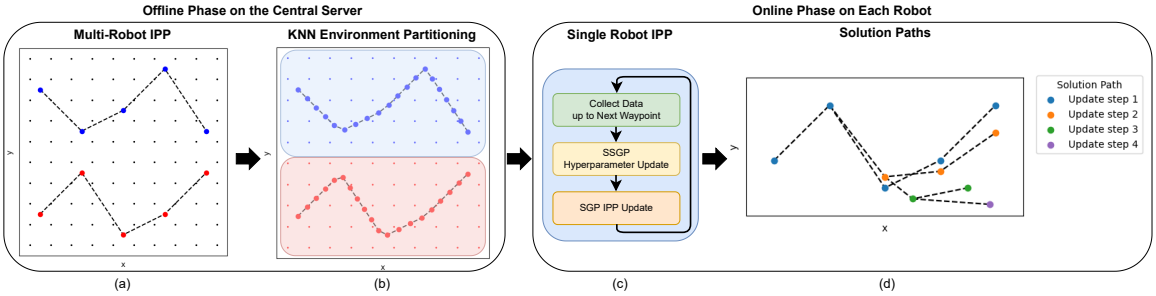


Figure 4.1: Overview of our online decentralized IPP approach for two continuous sensing robots. (a) Our approach first solves the multi-robot IPP problem offline with random hyperparameters. (b) We next use KNN to partition the environment among the robots. (c) For each robot, we then iterate between hyperparameter updates using an SSGP and IPP solution path updates using an SGP. (d) Illustration of the solution path updates for a single robot.

4.4.2 Decentralized Online Updates

We address the decentralization problem by leveraging a warm start approach to multi-robot IPP and then partitioning the search space into monitoring regions for each robot. Specifically, we first solve the multi-robot IPP problem using our offline IPP approach (Algorithm 4). Since we do not assume access to any information about the data field in the environment, we optimize the offline IPP solution using random hyperparameters.

Next, we partition the environment into regions for each robot to monitor. We do this by leveraging the K-nearest neighbor (KNN) classifier [3] on the waypoints

corresponding to the offline IPP solution path of each robot. The waypoint locations serve as the inputs $\mathbf{X}^{\text{KNN}}$ of the training set, and the labels $\mathbf{y}_{\text{KNN}}$ are set to the index of the robot path to which the waypoints correspond. Also, we interpolate additional waypoints along the offline solution's original s waypoints and add them to the KNN's training set to ensure that the KNN classifier accurately partitions the environment.

We then utilize the KNN classifier to predict labels for the $\mathbf{X}^{\text{environment}}$ random unlabeled samples used to optimize the offline SGP-based IPP approach. The pseudocode of the method is shown in Algorithm 7. It is important to note that the predicted labels here correspond not to the value of the phenomenon being monitored, but rather to the index of the robot to which that point in the environment is assigned. Moreover, the KNN classifier has a constant training time of $\mathcal{O}(1)$ since it only needs to store the training samples, and prediction can be efficiently carried out in $\mathcal{O}(nd)$ time, where n is the number of training samples and d is the dimensionality of the data.

Algorithm 7: Offline Phase of the decentralized IPP approach run on the central server.

Input: Domain of $\mathcal{V}$, number of waypoints s, number of robots r, path constraints $\mathbf{C}$

Output: Monitoring regions $\mathbf{X}^{\text{regions}}$, initial solution paths $\mathcal{P}$, hyperparameters θ

$\theta \sim \mathcal{U}(.)$ // Draw from a uniform distribution

$\mathcal{P}, \mathbf{X}^{\text{environment}} = \text{SGP-IPP}(\theta, \mathcal{V}, s, r, \mathbf{C})$

for $i \leftarrow 1$ **to** r **do**

  $\mathbf{X}_i^{\text{KNN}} = \text{Upsample}(\mathcal{P}_i)$

  $\mathbf{y}_i^{\text{KNN}} = \{i | x \in \mathbf{X}_i^{\text{KNN}}\}$

$\mathbf{X}^{\text{KNN}} = \cup_{i=1}^{r} \mathbf{X}_i^{\text{KNN}}$; $\mathbf{y}^{\text{KNN}} = \cup_{i=1}^{r} \mathbf{y}_i^{\text{KNN}}$

$\mathbf{y}^{\text{regions}} = \text{KNN}(\mathbf{X}^{\text{environment}}; \mathbf{X}^{\text{KNN}}, \mathbf{y}^{\text{KNN}})$

for $i \leftarrow 1$ **to** r **do**

  $\mathbf{X}_i^{\text{regions}} = \{x \in \mathbf{X}^{\text{environment}} | \mathbf{y}^{\text{regions}}(x) == i\}$

return $\mathbf{X}^{\text{regions}}, \mathcal{P}, \theta$

Once we have the labels determining which regions of the environment are assigned to each of the r robots, we send each robot its initial offline IPP solution path along

Algorithm 8: Online Phase of the decentralized IPP approach run on each robot.

Input: Robot i monitoring regions $\mathbf{X}_i^{\text{regions}}$, initial solution path $\mathcal{P}_i$, path constraints $\mathbf{C}$, hyperparameters θ
Output: Data along the traversed path $\mathbf{X}^{\text{path}}, \mathbf{y}^{\text{path}}$

SGP-IPP.initialize($\theta, \mathbf{X}_i^{\text{regions}}, \mathbf{C}, \mathcal{P}_i$)
SSGP.initialize(θ)
$\mathbf{X}^{\text{path}} = \{\}, \mathbf{y}^{\text{path}} = \{\}$
for $j \leftarrow 1$ **to** $|\mathcal{P}_i|$ **do**
    $\mathbf{X}^{\text{batch}}, \mathbf{y}^{\text{batch}} = \text{CollectData}(\mathcal{P}_i[j])$
    $\theta = \text{SSGP.update}(\mathbf{X}^{\text{batch}}, \mathbf{y}^{\text{batch}})$
    $\mathcal{P}_i = \text{SGP-IPP.update}(\theta, j)$
    $\mathbf{X}^{\text{path}} = \mathbf{X}^{\text{path}} \cup \mathbf{X}^{\text{batch}}$; $\mathbf{y}^{\text{path}} = \mathbf{y}^{\text{path}} \cup \mathbf{y}^{\text{batch}}$
return $\mathbf{X}^{\text{path}}, \mathbf{y}^{\text{path}}$

with $\mathbf{X}_i^{\text{regions}}$, the subset of the random sampled locations assigned to the robot by the KNN classifier. Each robot then begins the online IPP phase. During this phase, the robots traverse along the current solution path, and once a specified number of data samples are collected, they use that batch of data to update the SSGP's hyperparameters. Subsequently, the updated hyperparameters are forwarded to the SGP-based IPP approach, which utilizes the new hyperparameters to (re)optimize the unvisited waypoints of the current solution path. Note that since the SGP-based IPP approach resumes optimizing the solution path starting from the previous iteration's solution path, its computation time is further reduced. Additionally, we disable gradient updates to the waypoints that have already been visited. Consequently, the online IPP method accounts for the distance traveled and ensures that the total distance budget is still satisfied by the updated solution path. The pseudocode for this online phase is shown in Algorithm 8.

This approach can adapt its solution path(s) to the information from the environment in a computationally efficient manner. Indeed, our centralized offline approach (Algorithm 4) has a runtime complexity of $\mathcal{O}(nm^2 + m^3)$, which is the SGP's complexity. Here, n is the number of randomly sampled unlabeled locations in $\mathbf{X}^{\text{environment}}$

and m is the number of SGP inducing points, which in our case is rs, i.e., the number of robots times the number of waypoints for each robot. In practice, we found that $n < 1000$ was sufficient for the SGP to approximate the extent of the environment well, and a larger n was needed only if there were complex obstacles or regions to be avoided in the environment. Moreover, the number of waypoints only controls the complexity of the solution path, for instance, how many turns it can contain. As such, one can even use a small number of waypoints to monitor a large area with minimal degradation in performance. Similarly, our online approach has a complexity of $\mathcal{O}(nm^2 + m^3 + \hat{n}\hat{m}^2 + \hat{m}^3)$. Here, $\hat{n}\hat{m}^2 + \hat{m}^3$ is the added SSGP training cost, with $\hat{n}$ as the data batch size at each update and $\hat{m}$ as the number of inducing points used in the SSGP. Finally, the decentralized approach has a similar complexity as our online approach but with n reduced to the number of points in the $\mathbf{X}_i^{\text{regions}}$ assigned to each robot and m reduced to just the number of waypoints of each robot. Thus, our approach is feasible even on a robot with minimal computational resources.

Our method can even handle heterogeneous robots. In the centralized case, we formulate the expansion transformation T_{exp} used in the SGP-based IPP approach to transform the set of inducing points corresponding to each robot to account for the robot's sensing model and update the SGP's optimization objective with constraints on those inducing points to model path constraints. In the decentralized case, since each robot uses its own SGP for IPP, we can use the same transformation-based approach for the robot's sensing model and path constraints. See [77] for further details.

4.5 Experiments

We demonstrate our IPP approaches on datasets related to ocean salinity, elevation, and soil moisture. The ROMS ocean salinity dataset [48] contains salinity data from the Southern California Bight region (USA), the elevation dataset [90] contains data from near the coast of North Carolina (USA), and the soil moisture dataset [47]

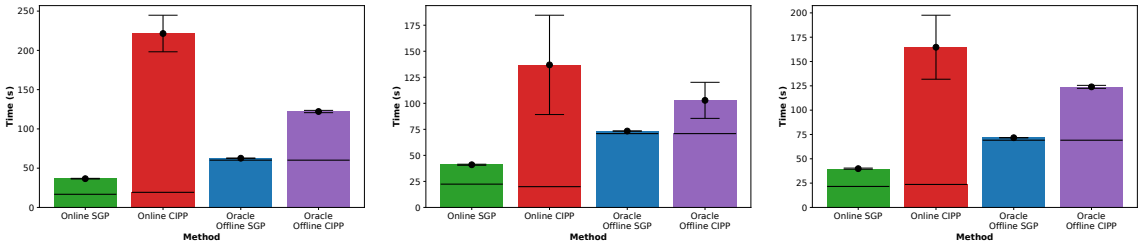
contains data from across the continental USA.

An RBF kernel [2] was used to model the spatial correlations of the datasets in all approaches. We evaluated the paths by gathering ground truth data along the generated solution paths (i.e., by continuous sensing robots) and estimating the state of the entire environment from the collected data using a GP with the ground truth kernel parameters of the dataset. The root-mean-square-error (RMSE) between the ground truth data and our estimates was used to quantify the solution paths. All benchmarks were repeated 10 times on a workstation with an 18-core Intel(R) Xeon(R) Gold 6154 3.00GHz CPU and 128 GB of RAM.

Single Robot: For the unconstrained online single-robot IPP problem, we benchmarked our SGP-based online IPP approach (Online SGP), which optimizes a path with s waypoints. The method assumes that no prior data from the environment is available and instead utilizes an SGP initialized with randomly sampled hyperparameters to generate the initial path. It then updates both hyperparameters and the path online, as detailed in Section 4.4. As a baseline, we also implemented the Continuous-Space Informative Path Planner (CIPP, Hitz et al. [36]). CIPP leverages CMA-ES, a genetic algorithm, to find informative paths maximizing mutual information in continuous spaces. We chose this method as a baseline because it is the most closely related approach capable of handling online multi-robot IPP in continuous spaces.

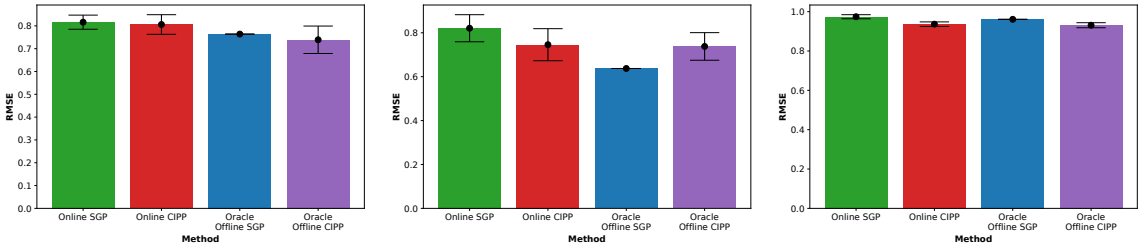
In particular, we implemented the online version of the method (Online CIPP), which also assumes that no prior data is available from the environment. However, Online CIPP employs a full GP to learn the hyperparameters; this does not scale well to large IPP missions due to the $\mathcal{O}(n^3)$ computational expense of full GPs and the limited computational resources available on most robot platforms. Therefore, to ensure a fair benchmark that focuses solely on the online IPP part of the algorithms, we modified Online CIPP with our SSGP-based hyperparameter learning approach

while retaining the original IPP part of the CIPP algorithm.



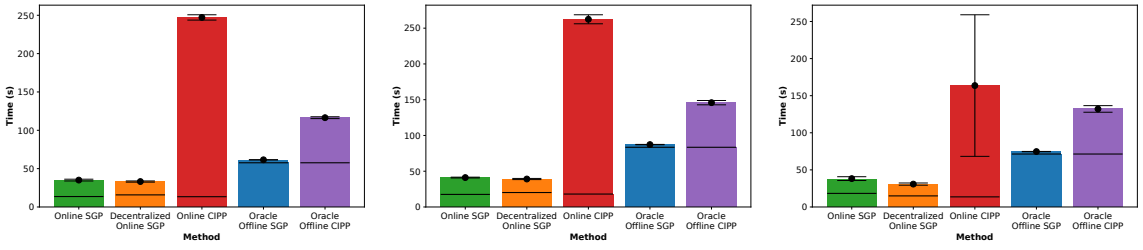
(a) Salinity dataset (b) Elevation dataset (c) Soil moisture dataset

Figure 4.2: The mean and standard deviation of the total runtime for single-robot IPP for each method on three datasets (lower is better). The hatched regions indicate the runtimes for the SSGP-based hyperparameter learning; for the offline oracle approaches, the hatched portion represents the full GP training time.



(a) Salinity dataset (b) Elevation dataset (c) Soil moisture dataset

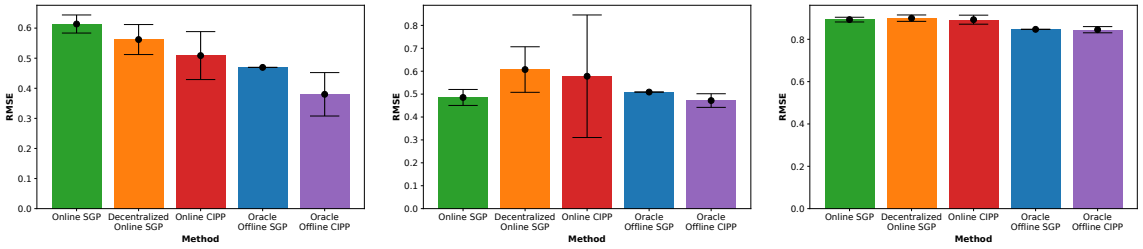
Figure 4.3: The mean and standard deviation of the RMSE scores for each single-robot IPP method on each of the datasets (lower is better).



(a) Salinity dataset (b) Elevation dataset (c) Soil moisture dataset

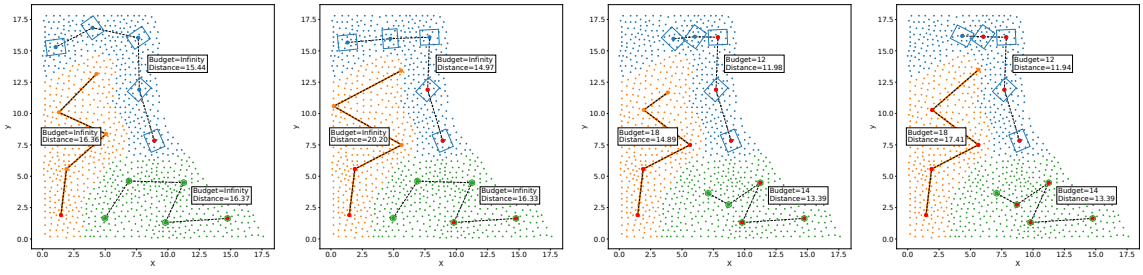
Figure 4.4: The mean and standard deviation of the multi-robot IPP runtime, with four robots, for each method on each of the datasets (lower is better).

Additionally, we benchmarked the offline versions of our SGP-based IPP approach [74] and the CIPP approach. The offline methods (Oracle Offline SGP and Oracle Offline CIPP) here assume they possess optimal hyperparameters that model the data field



(a) Salinity dataset (b) Elevation dataset (c) Soil moisture dataset

Figure 4.5: The mean and standard deviation of the RMSE scores for each multi-robot IPP method on each of the datasets (lower is better).



(a) Warm-start (b) Online update 1 (c) Online update 2 (d) Online update 3

Figure 4.6: Online and decentralized heterogeneous multi-robot IPP with three robots for ocean salinity monitoring. The robots modeled point sensing (green), continuous sensing (orange), and non-point rectangle FoV sensing (blue). Note that the monitoring regions are limited to the ocean; the top right portion of the map is Southern California and is not included in the monitoring regions. In (a) and (b), the paths were unconstrained. In (c) and (d), the paths were distance constrained. The red points indicate the waypoints that have been visited by the robots.

in the environment. Therefore, we first train a full GP with training data from the environment and utilize those hyperparameters in the offline approach. These methods are referred to as oracle approaches, as they have a priori information about the environment, which is not available to the online approaches.

All approaches were optimized for paths with 10 waypoints. The mean and standard deviation of the total runtimes for the IPP methods and the RMSE results are shown in Figure 4.2 and Figure 4.3, respectively. For completeness, we also indicate the runtimes for the SSGP-based hyperparameter learning part of the methods in the figures with the hatched regions. For the offline oracle approaches, the hatched regions represent the full GP training time. Note that the SSGP optimization time

varies depending on the number of data samples collected along the path, and the path is in turn affected by the parameters learned by the SSGP.

As we can observe, our Online SGP IPP approach is significantly faster than the Online CIPP IPP approach. Furthermore, we note that our approach's RMSE scores closely match those of the Online CIPP, Oracle Offline SGP, and Oracle Offline CIPP, despite having no initial training data from the environment. This demonstrates that the Online SGP approach can be deployed on compute-limited robots in novel environments while still achieving effective environmental monitoring performance.

Multiple Robots: Next, we demonstrate our online centralized and decentralized multi-robot IPP approaches (Online SGP and Decentralized Online SGP, respectively) on the three datasets. We also present results from the baselines: the online centralized multi-robot CIPP approach (Online CIPP) and the oracle offline centralized multi-robot versions of the CIPP and SGP IPP approaches (Oracle Offline SGP and Oracle Offline CIPP, respectively). All approaches in this experiment were optimized for 4 robots with 10 waypoints in each path. The resulting IPP runtimes and RMSE results (computed with data from all four robots) are shown in Figure 4.4 and Figure 4.5, respectively.

For the decentralized approach, we report the average runtime of the 4 robots, as unlike the centralized approach, the decentralized approach simultaneously runs online single-robot IPP on each robot. As we can see, our online centralized multi-robot and online decentralized multi-robot approaches are significantly faster than the baseline CIPP-based approaches while maintaining RMSE score performance. This demonstrates that our online decentralized multi-robot IPP approach can be deployed on compute-limited robot teams without stringent communication requirements, and still achieve good environmental monitoring results.

Heterogeneous Robots: We now demonstrate our decentralized online IPP approach for heterogeneous robots on the ocean salinity dataset. We used three robots:

one with point sensing, one with continuous sensing, and one with non-point rectangle FoV sensing. We generated the initial paths and environment partitioning assuming homogeneous robots, as detailed in Algorithm 7, and then deployed the three robots with online IPP, each with its corresponding sensor model as detailed in Algorithm 8. Upon reaching the third waypoint, each robot was subjected to a distance budget constraint. The resulting solution paths are shown in Figure 4.6. As we can see, the warm-start solutions cover most of the environment even though they were optimized with random hyperparameters. After the first online update with data from the environment, the paths are adjusted to account for the sensor FoV on each robot (continuous and non-point sensors in particular). Upon adding a different distance budget constraint on each robot, the paths are shortened. At the final leg, the paths are updated to fully utilize the distance budget. Moreover, the non-point rectangle sensing robot's FoV rotation angle was also treated as an optimizable variable. As such, we see that the final solution ensures that the FoVs are rotated to avoid overlapping and to gather the maximal amount of new data. Note that we modeled homogeneous robots in the offline phase to demonstrate the path updates for different sensors; our method can also model heterogeneous robots in the offline phase.

In the above experiment, the point sensing robot spent a total of 4.06 secs on SSGP optimization and 7.28 secs on IPP optimization. The continuous sensing robot spent 4.57 secs and 11.17 secs on SSGP and IPP optimization, respectively. The non-point rectangle FoV sensing robot spent 6.36 secs and 23.27 secs on SSGP and IPP optimization, respectively. The data from the three robots collectively resulted in an RMSE of 0.84. Note that the RMSE is higher compared to the benchmark results because we used a point sensing and non-point sensing robot here, while the benchmark modeled all robots as continuous sensing robots. This experiment demonstrates that our approach can effectively handle large-scale decentralized online IPP missions that require infinite horizon monitoring with a large number of heterogeneous robots.

4.6 Conclusion

This paper addressed the informative path planning (IPP) problem for environmental monitoring where the amount of data is continually increasing. Prior approaches, which leveraged GPs to update the hyperparameters online, would not scale well to real-world problems. Also, most earlier approaches used computationally expensive discrete optimization methods for IPP. We presented an online and decentralized heterogeneous multi-robot IPP approach. We tackled the online hyperparameter estimation issue by leveraging streaming sparse Gaussian processes and generalized our offline sparse Gaussian process-based IPP approach to handle online IPP. Moreover, we presented an efficient decentralized generalization of the online approach by warm-starting the multi-robot IPP solution and formulating a K-nearest neighbor-based environment partitioning scheme. We provided benchmarks on three real-world datasets demonstrating our approach's computational efficiency on the single robot and multi-robot IPP problems and compared it to a baseline genetic algorithm-based IPP approach. The IPP approach is also demonstrated with three heterogeneous robots with point sensing, continuous sensing, and non-point FoV sensing models, and distance budgets introduced halfway through the mission. Our future work will demonstrate our approach on real-world robots and investigate related problems such as IPP for 3D surface inspection. We additionally plan to generalize our approach to handle trajectory planning.

CHAPTER 5: Informative Path Planning in Graphs for Source Localization

5.1 Introduction

Methane accounted for 10% of the global greenhouse gas emissions in 2018 [91]. However, in the form of natural gas, methane is a viable energy source that can slow global emissions since it has a smaller carbon footprint than most other fossil fuels [92]. Unfortunately, it is not possible to extract without leaking, and its carbon footprint is small only if under 4% of its total production volume leaks [93]. Since methane leaks are unavoidable, we need to estimate leak rates in oil fields and take appropriate actions depending on the estimated rate.

Figure 5.1: Illustration of an oil field depicting storage tanks. Note that methane gas is not in the visible spectrum; it is shown green for visualization.

However, estimating the leak rates of gas sources is non-trivial. Recent work [94] has shown that current methods have heavily underestimated methane leaks. The

reason leak rate estimation is difficult is because it is inherently an ill-posed problem. Even if a single source is considered, multiple leak rates could result in sensing the same gas concentration at a given location as environmental factors such as wind speed and temperature could affect the dispersion of the gas. Furthermore, when multiple leak sources are considered (Figure 5.1), it is possible to have overlapping gas plumes making it difficult to attribute the data to each source.

Robots can collect gas concentration data from oil fields to estimate leak rates. The data collection locations have to be planned so that each location is highly informative and reachable within the robot's distance budget. So we also have to consider the informative path planning problem (IPP) for mobile robots. Even if we restrict data collection to a road network, depending on the size of the road network and distance budget, there could be a prohibitively large number of possible data collection walks[1].

We address two main problems in this chapter. First, we derive a computationally efficient probabilistic approach for estimating gas leak rates. We improve on the approach of Albertson et al. [95] by introducing a simplifying Gaussian assumption that results in substantial computational gains while retaining leak rate estimate convergence. Second, we address the IPP problem; we use the Generalized Cost-benefit (GCB) algorithm [96] to find data collection walks. However, the GCB algorithm does not consider arc routing constraints needed to find informative data collection walks in road networks; we introduce a variant that considers such constraints. We also present a modification to the GCB algorithm that substantially improves its runtime efficiency.

This chapter makes the following contributions:

1. Presents a fast and effective Bayesian approach for leak rate estimation from gas concentration data.

[1]A *walk* is any sequence of alternating vertices and edges $v_1, e_1, v_2, e_2, ..., v_k, e_k, v_{k+1}$ in a graph such that each edge e_i has endpoints v_i and v_{i+1}. A walk could contain repeated edges or vertices. It is considered closed if the first and last vertices are the same, and open otherwise.

2. Derives an efficient analytical solution to an information metric—Expected Entropy Reduction (EER)—that is used in the IPP problem.

3. Improves the runtime efficiency of the GCB algorithm used to solve the IPP problem.

4. Introduces an arc routing variant of the GCB algorithm for IPP in graph networks.

5.2 Problem Statement

We are given a road network graph $G = (V, E)$ with intersections modeled as vertices V and roads as edges E. We are also given a set of candidate leak locations and environmental factors such as wind speed and ambient temperature. We need to accurately estimate the leak rate of each leak source. The problem entails identifying informative data collection walks within a distance budget b and using the collected data to estimate the leak rates. Furthermore, depending on the leak rate of each source, we can detect the gas leaks at varying distances from the source. Therefore, our goal is to find minimal length data collection walks by selectively approaching the sources and getting only as close as needed to the sources. Additionally, the estimates could have high variance or become obsolete as environmental factors and leak rates continually change over time. Thus the solution approach would have to be fast, accurate, and iterative to update the leak rate estimates whenever needed.

5.3 Related Work

Leak rate estimation: There are several approaches to leak rate estimation based on whether the sensors are fixed or mobile, the type of gas concentration sensors, and resolution of the sensors.

Pandey et al. [97] showed that it is possible to estimate methane leak rates from satellite measurements. However, satellites are expensive to deploy and maintain.

The approach also has a limited ground pixel resolution and suffers occlusion from clouds, making it challenging to estimate small scale leaks omnipresent in oil fields.

An alternative to satellite data based sensing is to deploy a methane sensor at each oil well. To ascertain the feasibility of such a method, Project Astra [98] aims to build a sensor network for an entire oil field and develop a network monitoring method. However, it might be challenging to deploy and maintain a sensor network in an oil field like the Permian Basin in the US, spanning about 220,000 square kilometers with over 3500 drilled but uncompleted wells (DUC) [99], given the large number of sensors required.

Travis et al. [100] addressed methane leak rate estimation using fixed sensors by training a Neural Network (NN) on data from gas leak simulations. The NN could predict leak rates with reasonable accuracy but assumed stationary methane gas concentration sensors. Furthermore, the NN was an entirely data-driven, black-box approach that does not generalize to oil fields that do not follow the same leak rate distribution as the simulated data.

Albertson et al. [95] developed a Bayesian model for leak rate estimation in an oil field. The Generalized Extreme Value (GEV) Type II distribution [101] was used as the prior distribution in their approach. They also used Expected Entropy Reduction (EER) as an optimization metric to find data collection paths for mobile sensors. However, using the GEV distribution necessitated approximation methods like numerical quadrature to evaluate the nested integrals involved in the computation of EER and the posterior distribution of leak rates.

Furthermore, the framework of [95] is an iterative approach wherein one generates a data collection path, collects data, updates the leak rate estimates, and repeats the process until convergence to the true leak rates. However, every new iteration introduces an additional nested integral, each incurring substantial computation costs. Although the approach is theoretically elegant, it is computationally prohibitive and

infeasible for large oil fields.

Informative Path Planning: Finding the most informative walk for data collection is known as the Informative Path Planning (IPP) problem. Despite its name, IPP is not limited to just paths but includes tours and walks as well[2].

Usually, information metrics such as mutual information are used to quantify the informativeness of data collection locations in IPP. But the IPP problem is known to be NP-hard [62], and as a consequence, only suboptimal solutions can be obtained for most real-world problems.

Hollinger and Sukhatme [103] presented branch and bound techniques for IPP and established asymptotically optimal guarantees; their algorithms converge to the optimal solution as the run time approaches infinity.

A recent approach presented by Bottarelli et al. [66] developed active learning algorithms for the IPP problem with a complexity of $\mathcal{O}(|D|^5)$ where D is the discretized data collection space. They also suggested optimizations that trade search space complexity for reduced computation time.

Some IPP methods exploit structure in their optimization functions, mainly submodularity [104, 105, 106, 57], a property often found in information metrics used in IPP. Submodular functions have a diminishing returns property that makes them amenable to greedy optimization with known approximation factors. Iyer and Bilmes [107] established tight approximation factors for the maximization of submodular functions with submodular constraints.

Zhang and Vorobeychik [96] developed the Generalised Cost-benefit (GCB) algorithm with approximation guarantees to find a subset of vertices in a graph that maximize submodular functions with node routing constraints. They imposed node routing constraints by solving the Travelling Salesperson Problem (TSP) while ensuring that the walk is within the distance budget and includes the selected vertices

[2]A walk with no repeated edges is called a *tour*, and a walk with no repeated vertices is called a *path* [102].

of the graph. In practice, the method therefore requires numerous computations of the TSP, making it relatively expensive, and the method was limited to node routing constraints.

A closely related problem is gas distribution estimation [108, 109]. In this problem, gas leak sources are at unknown locations. The task is to estimate the gas density at each region and locate the leak. The problem also entails determining the data collection locations. Arain et al. [110] presented an approach that discretizes the search space and solves a convex relaxation of an integer linear program for near-optimal environment coverage.

Nonetheless, our problem is intrinsically different from the conventional IPP problem. We are interested in collecting data only to predict the leak rates of potential sources in an oil field instead of building a model of the entire data collection space. Therefore, an optimal walk for our problem might be substantially different from an optimal walk for the conventional IPP problem.

Albertson et al. [95] addressed our variant of the IPP problem by iterating over every possible path within the distance budget b between a given start and end location. The authors then computed each path's EER and selected the maximal EER path as the solution. However, a road network will have exponentially many possible walks, some of which might not even go near any leak source in the field. As such, the method incurs exponential computational costs to find the solution route and is feasible only for small road networks.

Arc Routing: Arc Routing Problems (ARP) [102] are closely related to TSP. But unlike node routing problems such as TSP that look for a walk that visits all nodes, ARPs look for a walk that traverses the arcs or edges of a graph at least once. The Rural Postman Problem (RPP) [111], a variant of ARP, is to find the shortest walk that traverses a specified subset of edges, the required edges of a graph. Since a walk needs to be continuous, RPP solvers may additionally use non-required edges.

5.4 Preliminaries

Foster-Wittig et al. [112] developed a gas dispersion model to calculate fugitive gas concentration at any location given the leak rate of the source. The gas concentration $C(s, x, y, z)$ at the location (x, y, z) when the leak rate is s is given by:

$$C(s,x,y,z) = \frac{s}{U}\left(\frac{\bar{A}}{\overline{z}(x)}exp\left[-\left(\frac{Bz}{\overline{z}(x)}\right)^2\right]\right) \times \left(\frac{1}{\sqrt{2\pi}\sigma_y}exp\left[-\frac{1}{2}\left(\frac{y}{\sigma_y}\right)^2\right]\right) \quad (5.1)$$

Here, U is the observed speed of the gas plume advection, $\bar{A}$, B, and $\overline{z}$ are functions of atmospheric stability, and σ_y is the length scale of the plume along the horizontal axis. The (x, y, z) coordinates are relative to the origin centered at the leak's source, with the x-axis along the wind direction.

Albertson et al. [95] used the above gas dispersion model in a Bayesian model to compute the posterior distribution of the leak rates given the methane gas concentration data from field measurements. A new instance of the Bayesian model was associated with each source.

The Bayesian model was also used to evaluate the Expected Entropy Reduction (EER) information metric φ to set up an optimization problem and find maximally informative data collection walks. EER measures mutual information [113], the amount of information one random variable contains about another. Mutual information can also be interpreted as the reduction in uncertainty of one variable due to the knowledge of another variable. Mutual information is treated as a dimensionless metric that can only be interpreted in its relative sense [113].

In [95], EER φ quantifies the information relevant to a source's leak rate in gas concentration data collected from a path. It also allows us to quantify this information without knowing the true leak rate. Here, M is the set of gas concentration data from the data collection path. Each $m \in M$ represents the measured gas concentration in

parts per million (ppm). $S \subseteq \mathcal{R}_{\geq 0}$ is the domain of the leak rate s.

$$\varphi[S; M] = -\log_2 \int_{s \epsilon S} p^2(s) ds + \int_{m \epsilon M} \log_2 \int_{s \epsilon S} \left(\frac{p(m|s)p(s)}{\int_{s_1 \epsilon S} p(m|s_1)p(s_1) ds_1} \right)^2 ds \; p(m) dm \quad (5.2)$$

EER is submodular [114]. Submodular functions are set functions with returns that diminish as the input set size increases. Any submodular function f satisfies the following property for sets X, Y, and T, with u being an element of the set T that is not already in Y.

$$f(X \cup \{u\}) - f(X) \geq f(Y \cup \{u\}) - f(Y)$$
$$\forall X \subseteq Y \subset T \;\text{ and }\; u \in T \backslash Y$$

Consider the EER function—adding more locations to a data collection path will increase the EER. However, the size of the incremental increases in the EER will diminish as the number of data points increases, as the amount of additional information in a path decreases with each newly collected data sample.

Like prior approaches, we assume that the oil field is on flat terrain without any large obstacles obstructing the gas plumes. Source detection is done by thresholding the leak rate of every well. Table 5.1 lists all the variables used in this chapter along with their definitions.

5.5 Approach

Our approach assumes a Gaussian prior for the leak rates, which we use to derive an analytical EER and posterior for the leak rates; we use the analytical EER to perform informative path planning.

Table 5.1: Definitions of variables.

Variable	Definition
$G = (V, E)$	Road network graph G, with vertices V and edges E
b	Distance budget
s	Leak rate of a source
S	Domain of leak rates
C	Gas dispersion model or gas concentration function
x, y, z	Coordinates along X, Y, and Z axes respectively
U	Observed speed of gas plume advection
$\hat{A}$, B, $\hat{z}$	Functions of atmospheric stability
σ_y	Length scale of gas plume along the Y axis
φ	Expected Entropy Reduction (EER)
M	Set of gas concentration data
m	Individual gas concentration data sample
$\mathcal{N}$	Normal distribution
μ_s	Mean of leak rate s of Gaussian prior
σ_s	Standard deviation of leak rate s of Gaussian prior
σ_e	Combined error of gas dispersion model and data measurement
$\mathcal{A}$	Function of all the terms except s in the gas dispersion model C
A_{xyz}	Scalar output of the function $\mathcal{A}$ at location x, y, z
$\mathbf{A}$	Vector containing A_{xyz} values computed at different locations
γ, μ, β	GEV Type II distribution parameters
$\hat{c}$	Routing problem solver (returns solution route cost)

5.5.1 Gaussian Assumption

To avoid the drawbacks of using the GEV Type II distribution as the prior over the leak rates, we instead use the Gaussian distribution as the prior. The Gaussian distribution is conjugate—if the likelihood is Gaussian, using a Gaussian prior over its mean will result in a Gaussian posterior. Moreover, its mean and variance compactly parameterize the distribution and are amenable to analytical computations.

The Gaussian prior assumption facilitates our derivation of an analytical equation to compute the EER and the posterior in time linear in the number of gas concentration samples collected from the field. Since Gaussian distributions have the maximal likelihood at the mean, instead of sampling the entire domain of s as was done in [95],

we only have to compute the mean to determine the most likely leak rate s (i.e., the maximum a posteriori probability estimate).

However, assuming a Gaussian distribution for the prior has its shortcomings. It is not consistent with the results of Brantley et al. [115] whose findings showed that the leak rates follow a log-normal distribution. Nevertheless, we found that the computational gains from computing both the EER and posterior analytically outweigh aligning the prior to a log-normal distribution. Moreover, our experiments show that our approach converges to the simulated leak rate despite the Gaussian prior. We next describe the critical steps in our derivations.

5.5.2 Analytical EER and Posterior

We formulate the prior $p(s)$ and likelihood $p(m|s)$ functions of the leak rate s as follows.

$$\begin{aligned} p(s) &\sim \mathcal{N}(\mu_s, \sigma_s^2) \\ p(m|s) &= \frac{1}{\sigma_e\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{m - C(s,x,y,z)}{\sigma_e}\right)^2\right] \end{aligned} \tag{5.3}$$

Here, μ_s and σ_s are the mean and standard deviation of the leak rate s. The combined gas dispersion model and concentration measurement error is σ_e. The gas concentration m is measured at coordinates x, y, z.

The evaluation of EER and the posterior involves integrating the likelihood function $p(m|s)$ with respect to s. However, the likelihood $p(m|s)$ contains the gas dispersion model C (Equation 5.1), which is a function of numerous parameters and seemingly intractable to analytical integration.

We found that the dispersion model C can be factorized into a product of s and the remaining terms independent of s. Therefore, we can combine all the terms other than s into a single function $\mathcal{A}$, dependent on the (x, y, z) coordinates where

the dispersion is calculated. This factorization allows us to treat the output of the function $\mathcal{A}(x, y, z)$ as a constant with respect to the likelihood $p(m|s)$, allowing us to treat the likelihood as a Gaussian. And since both the prior $p(s)$ and likelihood $p(m|s)$ are Gaussian, our posterior $p(s|m)$ will also follow a Gaussian distribution. We omit the arguments of $\mathcal{A}$ for brevity.

$$\begin{aligned} p(m|s) &= \frac{1}{\sigma_e\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{m - s\mathcal{A}(x, y, z)}{\sigma_e}\right)^2\right] \\ &= \mathcal{N}(s\mathcal{A}, \sigma_e^2) \end{aligned} \tag{5.4}$$

We derived[3] the EER φ and Gaussian posterior $p(s|M)$ using the factorized likelihood $p(m|s)$ and Gaussian prior $p(s)$. Here, M is a vector containing gas concentration data m at every sampling location. A_{xyz} is the scalar output of the function $\mathcal{A}$ computed for the sampling location with coordinates (x, y, z), $\mathbf{A}$ is the vector representation of all the A_{xyz} values, and c is a constant.

$$\begin{aligned} \varphi[S; M] &= -\log_2\left(\frac{1}{2\sigma_s\sqrt{\pi}}\right) + \sum_{m\in M} \frac{(m - \mu_s A_{xyz})^2}{2(A_{xyz}^2\sigma_s^2 + \sigma_e^2)} + c \\ p(s|M) &\propto \mathcal{N}\left(\frac{M^T\mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^T\mathbf{A}\sigma_s^2 + \sigma_e^2}, \frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^T\mathbf{A}\sigma_s^2 + \sigma_e^2}\right) \end{aligned} \tag{5.5}$$

Note that while computing the EER, M is calculated using a simulated leak source (Equation 5.1) with an arbitrary leak rate since we are only interested in the amount of information in a data collection walk, independent of the actual leak rate. However, to compute the posterior leak rate, we use gas concentration data collected from the generated data collection route.

Moreover, unlike the GEV prior model, the Gaussian prior model's posterior can

[3]The derivation and code can be found at
`https://github.com/UNCCharlotte-CS-Robotics/Gas-Leak-Estimation`

be represented by its mean and variance. So when the posterior needs to be used as the new prior distribution to determine subsequent data collection walks, it will not introduce any nested integrals as we can update the prior by changing its mean and variance.

5.5.3 Informative Path Planning (IPP)

Our approach to evaluating EER and the posterior substantially decreases the computation time. However, we still need to solve the IPP problem. Using the EER function as an optimization metric, we wish to find a data collection walk that maximizes the EER. Such a walk will result in the most informative sensor data for leak rate estimation.

We reformulate the problem as one where we have to find the edges in the graph G that maximize the aggregate EER and find a walk within the distance budget that includes all the selected edges. This problem, belonging to the class of arc routing problems with profits, is NP-hard [102].

Since EER is a submodular function, we could use the GCB algorithm [96] to maximize EER while imposing routing constraints. However, the GCB algorithm operates on nodes and imposes only node routing constraints. We need arc routing constraints, as the EER function operates on edges to quantify information. Solving the routing problem as a node routing problem does not always give us the best solutions. Furthermore, it nullifies the approximation guarantee established for the GCB algorithm [96].

We address this problem by proposing an ARP variant of the GCB algorithm. Our ARP variant selects edges of the graph instead of vertices and imposes arc routing constraints. We set up the routing problem as the Rural Postman Problem (RPP) [111] and solve it using an RPP solver. The RPP solver finds the shortest walk within the distance budget (if one exists) while including all the edges in the subset selected by the GCB algorithm. We found that the RPP variant of the GCB algorithm often re-

Algorithm 9: The modified Generalized Cost-benefit Algorithm (MGCB). $\hat{c}$ is the routing function: TSP when W is the node set V of the graph G, and ARP when W is the edge set E. φ is the submodular cost function (EER), and $W'\backslash x^*$ is the set W' without the element x^*.

Data: $b > 0, W$
Result: Walk $S \subset W$
1 $A := \arg\max\{\varphi(x) \mid x \in W, \hat{c}(x) \leq b\}$
2 $Z := \emptyset$
3 $W' := W$
4 **while** $W' \neq \emptyset$ **do**
5 **for** $x \in W'$ **do**
6 $\Delta_{\varphi}^{x} := \varphi(Z \cup x) - \varphi(Z)$
7 $\Delta_{c}^{x} := \hat{c}(Z \cup x) - \hat{c}(Z)$
8 **end**
9 $Y := \{x \mid x \in W', \hat{c}\,(Z \cup x) \leq b\}$
10 **if** $|Y| == 0$ **then**
11 **break**
12 **end**
13 $x^* = \arg\max\{\Delta_{\varphi}^{x}/\Delta_{c}^{x} \mid x \in Y\}$
14 $Z := Z \cup x^*$
15 $W' := W'\backslash x^*$
16 **end**
17 **return** $\arg\max_{S \in \{A,Z\}} f(S)$

sults in walks with higher or equivalent EER than those generated using the original TSP variant. Furthermore, the RPP variant retains the approximation guarantee of the GCB algorithm since both the subset selection and routing constraints operate on the edges of the graph.

Moreover, we also improve the runtime efficiency of the GCB algorithm by adding a conditional break statement (Algorithm 9). Let $S \subseteq W$ be the solution set generated by the GCB algorithm. The original algorithm takes $|W|$ iterations of the while loop. In contrast, our modified GCB algorithm (MGCB) takes only $|S|+1$ iterations of the while loop and returns the same result as the original algorithm.

The MGCB algorithm (Algorithm 9) starts by ensuring that at least one element (nodes if using TSP or edges if using ARP) is reachable within the distance budget (Line 1). Then it computes the increments in the route cost Δ_{c}^{x} and submodular

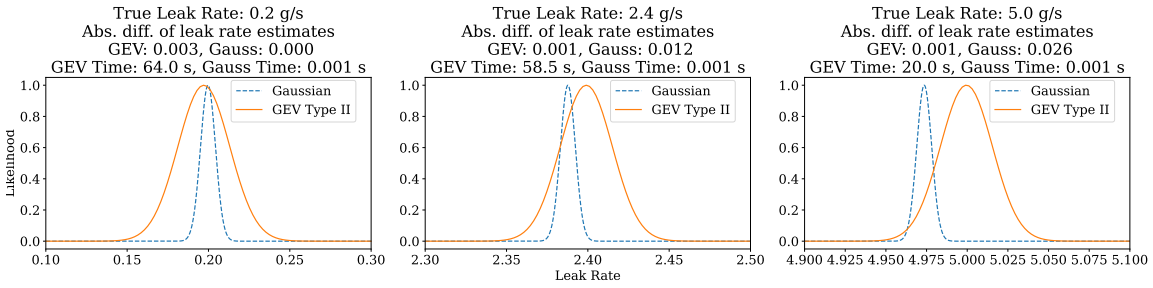


Figure 5.2: Posterior leak rate distribution with GEV Type II and Gaussian priors for different true leak rates. The posterior with GEV prior is shown in orange (solid line), and the posterior with Gaussian prior is shown in blue (dashed line). Subplot titles show the true leak rate, the absolute difference between each distribution's mode and the true leak rate, and the computation time. The GEV Type II and Gaussian priors were parametrized with a mode (most likely leak rate) of 0.09 g/s and 0.15 g/s respectively.

function cost Δ_{φ}^{x} upon adding each available element $x \in W'$ to the solution set Z (Lines 5-8). Any infeasible routes are filtered, and it checks if any feasible elements remain that it could add to the solution route (Lines 9-12). If none remain, the algorithm returns the best-known solution up to that point. Else it adds the element x^* with the highest increment ratio (of submodular cost to route cost) to the solution set Z and removes it from the available elements set W' (Lines 13-15). The algorithm iterates until either the available elements set is empty or the condition for the break statement is satisfied (i.e., there are no more feasible elements).

5.6 Simulation Experiments

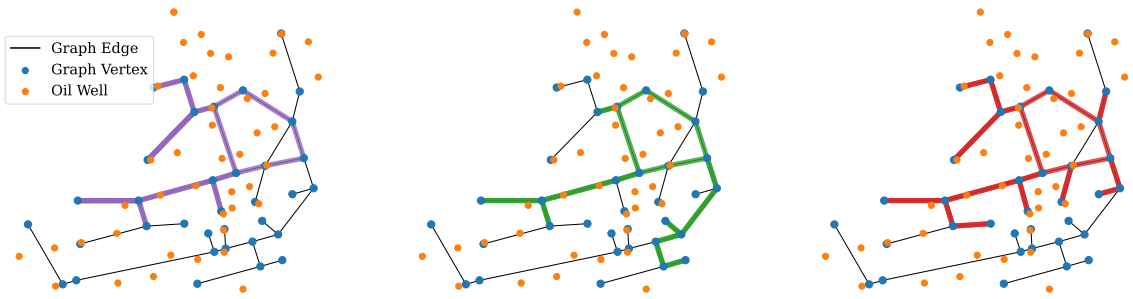
This section compares the EER and posterior computations using the GEV Type II and Gaussian priors. Additionally, it illustrates the advantages of the modified GCB algorithm for IPP. Real-world experiments on gas leak rates are stochastic, influenced by environmental conditions such as wind or humidity. To control for such variations, we conducted our experiments in simulation with the fixed environmental conditions documented in [116] and used the dispersion model (Equation 5.1) from Foster-Wittig et al. [112].

In all our experiments, we used the GEV Type II prior with model parameters

γ, μ, β, and σ_e set to 1, 0.19, 0.23, and 0.01 respectively, obtained from [95], and the Gaussian prior with μ_s, σ_s, and σ_e set to $0.15, 0.65$, and 0.03 respectively. The Gaussian prior's mean was estimated by fitting to data sampled from the GEV Type II distribution with the parameter values mentioned above. We can generalize this fitting process to any oil field of interest by replacing the GEV distribution data with aggregate historical leak data from that oil field. The σ_s and σ_e parameters were tuned so that the largest leak rate in our sampled data used to fit the Gaussian mean was within two standard deviations, which gave us Gaussian posterior predictions close to that of the original GEV prior model.

5.6.1 Leak Rate Posterior

First, to establish our approach's validity, we simulated a source leaking at three different leak rates and calculated gas concentrations M at ten random locations around the source. Using the GEV Type II and Gaussian prior-based approaches, we then used the simulated gas concentration data M to estimate the true leak rate s. The results are shown in Figure 5.2.



(a) Path iteration (1.80E-5 MSE)

(b) Modified GCB with TSP (2.20E-5 MSE)

(c) Modified GCB with RPP (1.08E-5 MSE)

Figure 5.3: Example graph extracted from the oil well corpus and the walk generated with each algorithm. The graph has 35 vertices, 36 edges, and 43 oil wells.

We found that using either prior we can accurately estimate the true leak rate. Even when the leak rate has a low likelihood in the prior distribution, such as 5 g/s, our approach's estimates are close to the true leak rate. But the Gaussian prior

based model underestimates the leak rate as the true leak rate moves far away from the prior distribution's mean.

However, this is a degenerate scenario, as we limited the data to only 10 points to show our approach's limits. Moreover, gas concentration sensors have much higher sampling rates, and according to Brantley et al. [115], most leaks occur at much lower rates, around 0.16 g/s, which is close to the mean μ_s of our prior distribution $p(s)$. As such, we anticipate that our approach's estimated leak rate would be closer to the true leak rate in a real-world scenario.

Furthermore, the Gaussian prior based approach is five orders of magnitude faster than the GEV prior based approach. This is because, unlike the GEV based approach, we can analytically compute our model's posterior.

5.6.2 Expected Entropy Reduction (EER)

The following experiment details the computational gains of our approach when computing the EER. We are interested in paths with the most information about leak rates quantified by EER. The EER computation cost is substantial when using the GEV Type II prior. We demonstrate the advantages of the Gaussian prior based approach to EER computation by evaluating the EER of six paths and sorting them in decreasing order. The EER values are not interpretable by themselves. Therefore, we are only interested in the order of the paths sorted by EER. We considered straight-line paths (each with 100 evenly sampled data points) parallel to the Y-axis at different distances along the X-axis from a simulated leak source.

We found that using either prior distribution to compute the EER gives the same ordering of paths. However, the Gaussian prior based model took five orders of magnitude less computation time compared to the GEV model, as shown in Table 5.2. Furthermore, our approach's benefits multiply in an actual oil field where hundreds of leak sources are considered, and the computation is repeated for numerous paths. We also observed fluctuations in the GEV model's computation time due to the stochastic

Table 5.2: EER computation time with GEV Type II and Gaussian priors for paths at varying distances from the leak source. The results were averaged over 10 iterations.

Distance to oil well along X-axis (meters)	Compute Time (secs)	
	GEV Type II prior	Gaussian prior
0.2	22.42391	0.00017
0.8	16.90775	0.00016
1.4	30.65850	0.00017
2.4	41.51963	0.00021
3.0	42.78649	0.00017
5.0	41.21629	0.00020

nature of adaptive quadrature used to evaluate the integrals in its EER. In contrast, the Gaussian prior model takes an almost constant amount of time to compute the EER, given the analytical solution to its integrals.

Our results above establish that our method converges to the true simulated leak rate with significantly reduced computation time despite the Gaussian assumption.

5.6.3 Informative Path Planning (IPP)

We also improved the IPP approach as mentioned in Section 5.5.3. The following experiment establishes the improvement in computation time of the IPP approach using our modified GCB algorithm.

We considered a corpus of 80,000 oil wells in the Permian basin in Texas, USA [117]. The wells were clustered into 1000 clusters based on their relative positions. We then extracted the road network associated with each cluster. To ensure that the path iteration algorithm's runtime is feasible, we empirically chose the connectivity range and total graph distance. We filtered out graphs with average node connectivity [118] higher than 1.1 and graphs with total road network length less than two times the distance budget b, which gave us 134 graphs. The statistics of the considered graphs are shown in Table 5.3.

We then generated data collection walks with a distance budget of 15 km, which we found to be the range feasible for selected unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs). We generated the walks using the path iteration

Table 5.3: Statistics of the graphs used to benchmark the MGCB algorithm.

Statistic	Min	Max	Mean
Number of oil wells	11	145	35.50
Number of vertices	15	420	76.28
Number of edges	16	484	81.07
Avg. connectivity [118]	1.0	1.10	1.04

approach of Albertson et al. [95], the GCB algorithm with TSP, our modified GCB algorithm with TSP, the GCB algorithm with RPP, and the modified GCB algorithm with RPP. The TSP [75] and RPP [111] [4] solvers we used had a 3/2-approximation guarantee. To compute the EER, we sampled the gas concentration at ten evenly spaced points along each edge in the route generated by the routing algorithm.

Table 5.4: Leak rate prediction mean squared error (MSE) and computation time for each method (lower is better). Path Iter is the path iteration, GCB is the original GCB algorithm, MGCB is our modified GCB algorithm, and the TSP/RPP postfix refers to the routing constraint solver used in GCB.

Method	Mean MSE	Std. MSE	Mean Time (secs)	Std. Time (secs)
Path Iter	1.6132E-01	1.2989E+00	1203.81	4.18
GCB TSP	2.0583E-04	1.4948E-03	1209.58	22.64
GCB RPP	2.5680E-05	1.3238E-04	1211.50	17.45
MGCB TSP	2.0583E-04	1.4948E-03	129.41	305.48
MGCB RPP	**2.5680E-05**	1.3238E-04	450.02	512.24

In this experiment, we used the Gaussian prior based EER computation method as it would be far too expensive to compute EER with the GEV prior. We randomly sampled the oil well leak rates from a uniform distribution over the range 0 to 6 g/s and assumed that we were given no prior leak rate estimates. Therefore we used the default μ_s and σ_s obtained from historical leak rate data [95]. Furthermore, we allocated a maximum of 20 mins to each algorithm for each graph. The mean squared error (MSE) of the leak estimates for each method and the computation times are shown in Table 5.4. Additionally, one of the generated graphs, along with its walks, is shown in Figure 5.3.

[4] We used the Line Coverage Library available at
`https://github.com/UNCCharlotte-CS-Robotics/LineCoverage-library`

Path iteration is infeasible for large graphs as the number of possible walks increases exponentially with the number of graph edges; this is reflected in its computation time. In almost all cases, it terminates with a timeout and returns the best-known solution up to that point, which also explains the low standard deviation of computation time.

In contrast, the GCB and MGCB algorithms' computational complexity does not grow exponentially. But GCB is still costly to compute and usually terminates with a timeout and returns the best-known solution up to that point, thereby underutilizing the distance budget. This also explains the low standard deviation of GCB's computation time.

Even though GCB finds the solution early on in its computation, it continues to iterate through the algorithm as there is no test to detect convergence and terminate the algorithm. However, MGCB converges to the same solution as the GCB algorithm in a fraction of the time on all considered graphs.

We notice a higher standard deviation in the computation time of MGCB because the graphs are of varying sizes, therefore taking a varying amount of time to solve. Finally, we also note that the RPP variant of GCB performs better than the TSP variant as the arc routing constraints align with the EER function that measures information along the edges of the graph.

5.7 Discussion

Suppose the routing constraint solver used in MGCB is stochastic, which is sometimes the case when using heuristics to solve the TSP/ARP. In this case, one should expect to see the original GCB algorithm improve its solution even after $|S|+1$ iterations of the algorithm. This is because even though the GCB algorithm converges to the solution subset of nodes/edges in $|S|+1$ iterations, the algorithm keeps solving the routing problem with the same required solution subset for $|W|-|S|-1$ iterations. Heuristic-based solvers usually find better solutions after a few such iterations. However, one could always use an exact solver once MGCB converges to improve the

solution and get similar results.

Also, note that even though the MGCB algorithm takes $|S|+1$ while loop iterations, $|S|$ could still be close to $|W|$ in the worst case. This would be the case when the distance budget b is large enough, and the total distance of the graph edges is small enough that the solution could traverse the whole graph. Nonetheless, our approach does not incur any significant additional computation costs. As such, it results in the same computation time as the original GCB algorithm.

A limitation of our work is that our experiments were conducted only in simulation. However, we did not change the gas dispersion model, which is the only component influenced by real-world conditions. Since the validity of the dispersion model and the Bayesian approach was already established by Albertson et al. [95], we believe field experiments would be consistent with our simulations.

Additionally, the dispersion model we considered assumed flat terrain and steady state environmental conditions. One could potentially develop a more sophisticated dispersion model that can be factorized into $\mathcal{A}$ and s to handle dynamic environmental conditions and use it in our approach.

5.8 Conclusion

We presented a method for efficient and accurate gas leak rate estimation of greenhouse gases such as methane. We derived a closed-form equation for EER, a mutual information metric, and substantially improved the runtime efficiency of the GCB algorithm used to maximize the EER to find informative data collection walks. Since the GCB algorithm did not consider arc routing constraints, we presented a GCB variant that addressed such constraints. We also derived an efficient analytical approach for computing the posterior distribution of the gas leak rate for each leak source.

Our simulated experiments, using oil well data, established the convergence of our approach to the true leak rate. We also showed that our approach computes the

EER and posterior leak rates five orders of magnitude faster than the prior approach. Furthermore, our modified GCB algorithm (MGCB) was shown to be at least an order of magnitude faster than the original GCB algorithm. Finally, we demonstrated that our ARP variant of the MGCB algorithm obtains data collection walks for oil fields that on average result in more accurate leak rate estimates when compared to the original GCB algorithm.

The TSP/ARP routing algorithm is invoked numerous times, with only one new element added to the required set in each iteration. One could reduce this computation cost by incorporating incremental solutions in each iteration. We plan to address this in our future work.

CHAPTER 6: Sensor Placement in Graphs for Source Localization

6.1 Introduction

There is growing evidence that the frequency of pandemics, such as the COVID-19 pandemic, is increasing [119, 120]. It is, therefore, crucial to identify infected individuals before they show symptoms and treat them to curb the spread of such outbreaks. Wastewater-based epidemiology has proven to be an effective approach for detecting viral and bacterial outbreaks ([121, 122, 123, 124, 125]) including influenza, poliovirus, respiratory syncytial virus, and Escherichia coli infections.

Figure 6.1: Illustration of a wastewater network with wastewater autosamplers.

Recent studies have demonstrated that viruses can be detected in human waste days before an infected individual exhibits symptoms [126]. Regular monitoring of wastewater networks for active traces of harmful pathogens can aid in identifying outbreak locations and their scale. A few case studies ([127, 128, 129, 130, 131, 132])

have shown that monitoring wastewater networks is a reliable and non-intrusive way to detect COVID-19 outbreaks.

Although autosamplers can automate the collection of wastewater samples, they can be costly. Additionally, the sampled wastewater must be collected and processed in a lab. As such, to keep operating costs in check for wastewater monitoring, it is crucial to minimize the number of autosamplers while ensuring accurate virus source localization and comprehensive population coverage. It is also challenging to accurately detect viruses in diluted wastewater samples. Thus, we must account for the concentration of the collected wastewater samples at the solution placement locations when using autosamplers.

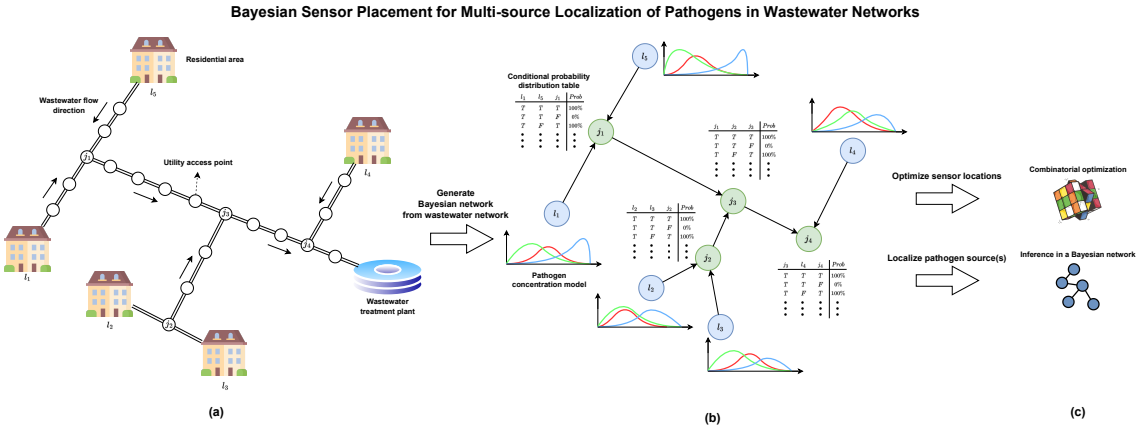


Figure 6.2: Illustration of our approach. (a) We consider wastewater networks and (b) generate a Bayesian graph from the wastewater network graph. (c) We then optimize the sensor placements using combinatorial optimization on the Bayesian graph and localize the virus source(s) using inference in the Bayesian graph.

We model the placement of autosamplers as a sensor placement problem. Our method leverages the underlying graph structure of wastewater networks to set up an optimization objective based on Bayesian networks. We use discrete optimization to obtain sensor placement solutions efficiently while also ensuring that the placements provide high-concentration wastewater samples from the autosamplers. Moreover, we introduce a graph Bayesian approach to localize multiple simultaneous virus outbreaks. To validate our approach, we conduct simulation experiments based on a

real-world wastewater network and demonstrate its accuracy and robustness. Our contributions are summarized below:

1. Introduce an approach to model a reduced graph representation of wastewater networks ideal for efficient sensor placement optimization and source localization.

2. Present an approach to use the inherent structure of wastewater networks to build Bayesian networks.

3. Present a computationally efficient graph Bayesian virus source localization approach that uses wastewater sample test results.

4. Establish a graph Bayesian optimization objective that can be used to find ideal sensor placements for accurate virus source localization. The method uses our source localization approach to ensure that the placements collect high-concentration wastewater samples.

6.2 Problem Statement

We are given a wastewater network modeled as a graph $G = (V, E)$ with buildings and utility access points[1] modeled as vertices V and pipelines as directed edges E whose direction is identified by the wastewater flow. Wastewater networks have a (reverse) directed tree structure with wastewater flowing from the leaf nodes (i.e., the buildings) to the root node (i.e., the wastewater treatment plant). Given their inherent graph structure, only buildings are associated with leaf nodes $l \in L$, and only utility access points are associated with non-leaf nodes $j \in J$ in the graph. We use upper case letters to represent a set of nodes, lower case letters to refer to the nodes in a set, and subscripts to indicate a specific node.

[1] Also referred to as maintenance holes, cleanouts, and sewer holes.

We are given k sensors (i.e., the wastewater autosamplers) that can be deployed to monitor the wastewater network. We must identify a set of nodes $W \subset V$ to monitor, i.e., select autosampler placements to accurately detect and localize virus outbreaks at one or more buildings in the wastewater network. Note that in the real world, the sensing nodes only collect the wastewater, which is then sent to a lab to be analyzed and determine if the samples are positive or negative for any viruses. Additionally, when wastewater from multiple sources is combined in pipelines, the wastewater is diluted, resulting in inaccurate virus detection test results. Therefore, we must also account for any sample concentration requirements on the wastewater while determining the solution autosampler placements.

We define a virus outbreak as a positive test for the virus of interest from the wastewater of any building. Deploying sensors at every building in the wastewater network would make detecting and localizing virus outbreaks a trivial task. However, as we are limited to only k sensors (where $k << |L|$), the problem of optimizing their placement is NP-hard [133], making it challenging to solve.

6.3 Approach

We first describe our method for reducing the wastewater network graph. Then, we explain how we construct a Bayesian graph from the reduced graph. Next, we present our approach for using the Bayesian graph to localize virus sources. Finally, we detail our method for sensor placement. Figure 6.2 shows an illustration of our approach.

6.3.1 Wastewater Network Graph Reduction

An inherent property of wastewater networks is that a virus outbreak can occur only at the graph's leaf nodes (i.e., the buildings). Furthermore, since the wastewater network graph has a (reverse) directed tree structure, we can leverage these two properties to reduce the number of nodes in the graph without losing any significant

information.

Indeed, we can exclude any non-leaf node with a single parent node, as shown in Figure 6.3. This is because placing a sensor at either the current node j_2 or its parent node j_1 would result in sensing the same wastewater that flows into node j_1 from its parent nodes (l_1 and l_2). Since choosing either node (j_1 or j_2) results in the same solution, and there is no benefit from sensing at both the nodes, we discard the current node j_2, which is a non-leaf node with a single parent. Note that after discarding the node j_2, we connect the parent node j_1 to the child node(s) of the discarded node j_2.

Such a reduction in the graph size reduces the size of the optimization problem that needs to be solved to find the ideal sensor placements without diminishing the final solution quality. Also, it would reduce the computation cost of our source localization approach.

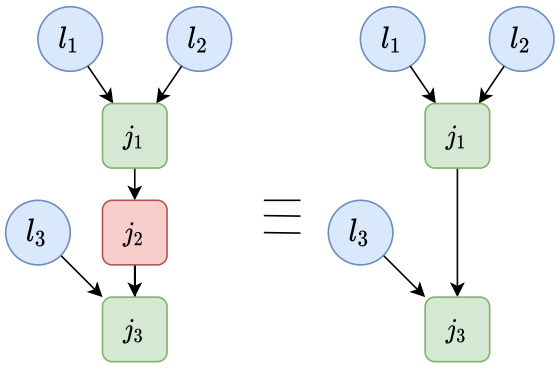


Figure 6.3: Node elimination scheme to reduce the graph size. Nodes with a single parent node are removed from the graph. Nodes in blue represent leaf nodes, green nodes are non-leaf nodes, and red nodes are nodes that we can remove from the graph.

6.3.2 Bayesian Graph Construction

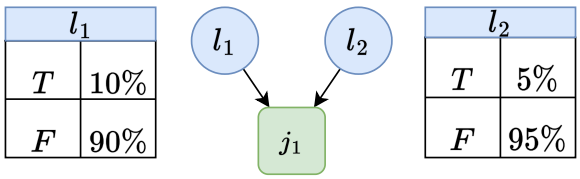
One of the main insights of this article is that we can utilize the inherent graph structure of wastewater networks to construct Bayesian graph networks. Bayesian graph networks incorporate random variables associated with each node in the graph and utilize the graph structure to model causal relationships among the random

variables. We can leverage Bayesian networks to calculate conditional probabilities that can be employed to establish an optimization objective for sensor placement and also predict the most probable source(s) of a virus outbreak.

We build a Bayesian graph B from the wastewater network graph G. In the Bayesian graph, we associate a Boolean random variable b with each node. The value of b indicates whether the wastewater flowing through that node contains viruses or not, where $True$ indicates the presence of viruses and $False$ indicates their absence. To construct the Bayesian graph, we must also define the distribution associated with each random variable. For the leaf nodes $l \in L$, the Boolean random variables are treated as Bernoulli distributed variables, and we parametrize the distributions to model the probability of a virus outbreak at each building.

The random variables b at the non-leaf nodes $j \in J$ are computed using a deterministic OR-gate operation over the random variables of their parent nodes. Although the junction nodes $j \in J$ in real-world wastewater networks might not behave as OR-gates, this simplifying assumption allows us to efficiently compute the conditionals using variable elimination and message passing techniques [134]. Indeed, given our directed tree graph structure and Boolean random variables, our Bayesian graph network is similar to the Noisy-OR Bayesian network [134], which is amenable to efficient Bayesian inference. In addition, our experiments show that our approach works well despite our simplifying assumption.

We assign a Conditional Probability Density (CPD) table [134] to each non-leaf node to store the result of the OR-gate operation. The CPD table of each node is filled by iterating over all possible instantiations of the states of the current node and its parent nodes. Each instantiation is assigned a probability of 100% if the instantiation is possible with an OR-gate and set to 0% otherwise, as shown in Figure 6.4.



l_1	10%
T	
F	90%

l_2	5%
T	
F	95%

l_1	l_2	j_1	$Prob$
T	T	T	100%
T	T	F	0%
T	F	T	100%
T	F	F	0%
F	T	T	100%
F	T	F	0%
F	F	T	0%
F	F	F	100%

Figure 6.4: An example Bayesian graph B for a wastewater network with two buildings. The nodes are color-coded: blue for leaf nodes $l \in L$, and green for junction nodes $j \in J$. Each CPD table's header row color and text indicate the associated node, and subscripts indicate the node index. T and F represent $True$ and $False$, respectively.

6.3.3 Source Localization

We can now use our Bayesian graph B to localize the source of virus outbreaks. Assuming we already selected a set of nodes $W \subset V$ to deploy our sensors (i.e., the wastewater autosamplers), we can use Bayesian conditioning [134] to predict the virus outbreak state $A_l(W_b)$ of each leaf node $l \in L$ (i.e., each building) given the current virus presence state of the sensing nodes W_b:

$$A_l(W_b) = \begin{cases} True & \text{if } P(b_l = True | W_b) > 50\% \\ False & \text{otherwise} \end{cases}. \tag{6.1}$$

Our Bayesian graph B is informed about the structure of the wastewater network

from the directed graph G, and the prior virus outbreak statistics of each leaf node from the associated Bernoulli distributions. Therefore, evaluating the most likely state $A_l(W_b)$ of each building by conditioning on the current state of the sensor nodes W_b leverages all the information available to us. In addition, since our Bayesian model is a binary graph, our source localization approach is still computationally feasible.

6.3.4 Sensor Placement

Our sensor placement approach leverages the Bayesian graph B to formulate an optimization objective that maximizes the source localization accuracy for a given scenario S. This accuracy is defined as the fraction of buildings whose virus outbreak state is correctly predicted. In this context, a scenario S represents a hypothetical virus outbreak indicating which of the leaf nodes (i.e., buildings) in the wastewater network are currently experiencing an outbreak.

However, we anticipate that in the real world, most scenarios involve only a small fraction of buildings experiencing a virus outbreak at any given time. As a result, even if our model predicts that all buildings are virus-free, the accuracy may still be high, as only a few buildings experiencing an outbreak are mislabeled. Therefore, optimizing and evaluating our sensor placement solutions using accuracy alone may not be the most suitable approach. As a result, we also consider precision, recall, and F1 scores as alternative evaluation metrics to determine the best optimization metric in the experiments section. In the remainder of this article, we will refer to our optimization metric as the *score* function, implying that it can be any of the aforementioned metrics.

To ensure that our sensor placement solutions W are not biased towards a single virus outbreak scenario S, we sample multiple scenarios, each consisting of random samples from the Bernoulli distributions associated with the leaf nodes L. Each scenario consists of a random sample ($True/False$) from each of the leaf nodes. Optimizing the score function on the sampled scenarios ensures that our solution

sensor placements W account for the probability of a virus outbreak in each building. We refer to this objective as the *score objective*:

$$\underset{W \subset V, |W| \leq k}{\arg\max} \ \mathbb{E}_S \left[score(A(W_b), S) \right] . \tag{6.2}$$

Here, $A(W_b)$ represents our predicted virus outbreak state for all the buildings, and S represents a virus outbreak scenario. Note that we can optimize the sensor placements by directly maximizing the score function since we have a discrete combinatorial optimization problem. Such optimization problems do not require differentiable operations. However, the above optimization problem is NP-hard [133], making it challenging to find the globally optimal solution. As a result, we use the naive greedy algorithm [135] to find the solution sensor placements. The greedy algorithm selects one sensing node at a time until the cardinality constraint k is met. Each new sensing node is selected by computing the increments in the optimization objective upon adding each candidate node to the current solution set and selecting the node that results in the largest increment.

$$W \leftarrow W \cup \{ \underset{v \in V \setminus W}{\arg\max} \mathcal{F}(W \cup \{v\}) - \mathcal{F}(W) \} . \tag{6.3}$$

Here, $\mathcal{F}$ is the objective function (Equation 6.2). A drawback of optimizing the score objective is that it results in sensors placed only at the leaf nodes with the highest probability of an outbreak. Since we have a limited number of sensors, usually $k << |L|$, using such a solution entails ignoring outbreaks in buildings with a lower probability of an outbreak.

To address the issue mentioned above, we have added an indicator function to the objective. This function, known as the coverage indicator function $\mathbb{1}_{\text{cov}}$, returns a value of 1 if a scenario S can be detected with the current sensor placements W, and 0 if the scenario cannot be detected. To evaluate the indicator function, we check if

a path exists from the buildings experiencing a virus outbreak to the current sensor placements W. We refer to this objective as the Coverage objective, which can be formulated as follows:

$$\underset{W\subset V,|W|\leq k}{\arg\max}\ \mathbb{E}_S\left[score(A(W_b),S)+\mathbb{1}_{\text{cov}}(W,S)\right], \tag{6.4}$$

$$\mathbb{1}_{\text{cov}}(W,S)=\begin{cases}1 & \text{if scenario } S \text{ can be detected}\\ & \text{with sensors at } W\\ 0 & \text{otherwise}\end{cases}. \tag{6.5}$$

By optimizing the Coverage objective, we can obtain solutions with sensing nodes capable of detecting outbreaks even at buildings with a low outbreak probability, although with a reduced source localization accuracy.

Adding sensing nodes: The methods discussed so far have only considered scenarios where there are no preexisting sensors deployed in the wastewater network. However, in practice, one might want to add additional sensing nodes to improve the source localization accuracy. We can achieve this by maximizing the following objective function, where W_{curr} represents the set of preexisting nodes in the wastewater network, and $V\backslash W_{curr}$ represents the set of nodes in V that are not in W_{curr}:

$$\underset{\substack{W\subset V\backslash W_{curr},\\ |W_{curr}\cup W|\leq k}}{\arg\max}\ \mathbb{E}_S[score(A(\{W_{curr}\cup W\}_b),S)+\mathbb{1}_{\text{cov}}(W_{curr}\cup W,S)].$$

Removing sensing nodes: One might also want to remove a specified number of sensing nodes from a wastewater network. This can happen if we want to monitor fewer sensors to reduce costs. We can achieve this with our approach by maximizing

the following objective:

$$\underset{\substack{W \subset W_{curr}, \\ |W_{curr} \setminus W| \leq k}}{\arg\max} \; \mathbb{E}_S[score(A(\{W_{curr} \setminus W\}_b), S) + \mathbb{1}_{\text{cov}}(W_{curr} \setminus W, S)] \; .$$

Note that both the problems of adding and removing sensing nodes can be solved using the greedy algorithm. However, to remove samplers, the greedy algorithm needs to be modified. In this variant, the algorithm selects the node that contributes the *smallest* increment to the total objective in each iteration, unlike the sensor addition variant, which picks the node with the largest increment.

6.3.5 Concentration Requirements

The methods discussed above assume that sensing nodes always report the correct state of the nodes, i.e., whether the wastewater passing through them contains viruses. However, this may not be the case in the real world. The qRT-PCR test, commonly used to analyze wastewater samples for traces of viruses, is sensitive to the concentration of the samples. Therefore, it is crucial to ensure that the concentration, measured by the number of RNA virus copies per liter of wastewater, is above a minimum threshold to obtain reliable sensing results.

However, if one were to model all variables that influence the concentration of wastewater samples, such as the volume of wastewater and the number of virus copies, in the Bayesian graph B, the conditional probability $P(b_l|W)$ used for sensor placement and source localization would be computationally intractable. Therefore, we construct a separate auxiliary Bayesian graph C, which models the variables that affect virus concentration (as shown in Figure 6.5). The auxiliary graph C allows us to efficiently compute the virus concentration at any node in the wastewater network, given the virus concentration at the leaf nodes $l \in L$ that are experiencing an outbreak in a scenario S.

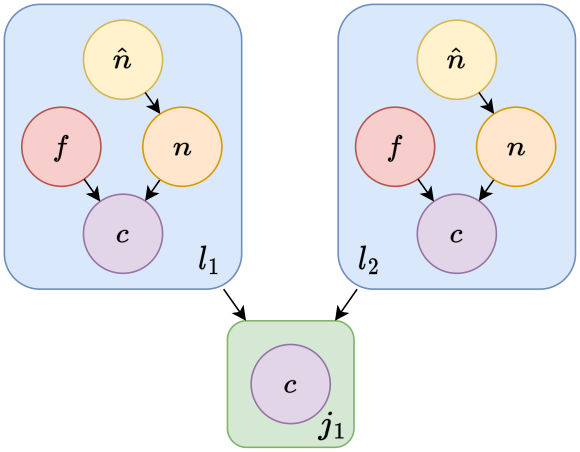


Figure 6.5: The concentration Bayesian graph C for an example wastewater network. The nodes are color-coded as follows, red nodes: wastewater flow volume f, yellow nodes: number of infected individuals $\hat{n}$, orange nodes: number of total virus copies n shed from each building, and purple nodes: wastewater concentration c. The blue and green nodes are the leaf and non-leaf nodes, respectively.

The auxiliary graph C is initialized with the wastewater network's graph structure similar to the binary Bayesian graph B. However, instead of using Boolean random variables to model each node's virus outbreak state, the auxiliary graph C employs multiple random variables to represent the virus concentration at each node. Our approach considers the volume of wastewater flow f, the number of infected individuals $\hat{n}$, and the total number of virus copies n for each building. Note that this approach can even use a more sophisticated concentration model.

We modeled the wastewater flow volume f of each building using a truncated Gaussian distribution, truncated at 0 to ensure that only positive flows are sampled. The number of infected individuals $\hat{n}$ was modeled using a Poisson distribution, while the number of total virus copies n at each building was computed by sampling the number of virus copies shed by each infected individual from a uniform distribution and then taking the sum. The wastewater virus concentration c_l at each leaf node $l \in L$ can be computed using the ratio of the total number of virus copies n to the wastewater flow volume f. To determine the concentration c_j at each non-leaf node $j \in J$, we use the conservation of mass equation shown below:

$$c_j = \frac{\sum_{i \in parent(j)} n_i}{\sum_{i \in parent(j)} f_i}. \tag{6.6}$$

The concentration rate c calculated at each node is then passed to an updated threshold indicator function $\mathbb{1}_{\text{thresh}}$, which enforces the minimum concentration requirement in our sensor placement optimization objective. We refer to this new optimization objective as the *thresholded coverage objective*.

$$\underset{W \subset V, |W| \leq k}{\arg\max} \; \mathbb{E}_S \left[score(A(W_b), S) + \mathbb{1}_{\text{thresh}}(W, S) \right] \tag{6.7}$$

$$\mathbb{1}_{\text{thresh}}(W, S) = \begin{cases} 1 & \text{if scenario } S \text{ can be detected with sensors} \\ & \text{at } W \text{ and every wastewater sample satisfies} \\ & \text{the concentration threshold} \\ 0 & \text{otherwise} \end{cases} \tag{6.8}$$

The threshold indicator function $\mathbb{1}_{\text{thresh}}$ (Eq. 6.8) is similar to the coverage indicator function $\mathbb{1}_{\text{cov}}$ (Eq. 6.5), but it also considers the concentration requirements for detecting a scenario. That is, even if a path exists from the buildings experiencing a virus outbreak to the current sensor placements W, the threshold indicator function $\mathbb{1}_{\text{thresh}}$ will return 1 (i.e., $True$) only if the concentrations at the sensing nodes that detect the scenario meet the required concentration threshold.

By optimizing the thresholded coverage objective, we can obtain sensor placements that not only enable accurate source localization but also ensure that the concentration of viruses in wastewater samples collected by the autosamplers is above the required threshold for reliable virus detection. It is worth noting that our approach still benefits from the computational efficiency of the Boolean OR-gate Bayesian network

B used to compute $P(S|W)$, while also accounting for the complex concentration requirements through the auxiliary Bayesian network C and Eq. 6.8. Thus, our approach overcomes the limitations of a computationally intractable Bayesian graph that models all variables together.

Note that the Bernoulli distributed random variable b_l in the Bayesian graph B, which indicates if an infection outbreak occurred at a building, is analogous to the Poisson distributed random variable $\hat{n}$ in the auxiliary Bayesian graph C, which represents the number of infected individuals. Therefore, when using the auxiliary Bayesian graph C to model the virus concentration, if we need to sample outbreak scenarios S to optimize our sensor placements, we sample them using the Poisson distributions associated with the random variable $\hat{n}$ instead of the Bernoulli distributions associated with the random variable b. But since the scenarios S sampled from Poisson distributions would indicate the number of infected individuals instead of the binary state of an outbreak, we apply the following *min* operation on each element of S, i.e., the sampled $\hat{n}$ at each building to ensure that the scenario is binary:

$$f(\hat{n}) = \min(1, \hat{n})\,.$$

Converting each scenario S into a binary vector enables us to use the binary Bayesian graph B to evaluate $P(b_l|W)$ for sensor placement and source localization. However, it is important to note that the threshold indicator function $\mathbb{1}_{\text{thresh}}$ is still computed using only the auxiliary Bayesian graph C. Additionally, when computing the wastewater concentrations, we sample the number of virus copies n at a leaf node l only if the corresponding building is indicated to be experiencing a virus outbreak in the scenario S. However, we always sample the wastewater flow volume f from all buildings to account for wastewater dilution. Table 6.1 lists all the variables used in this chapter along with their definitions.

Table 6.1: Definitions of variables.

Variable	Definition
$G = (V, E)$	Wastewater network graph G, with vertices V and edges E
B	OR-gate Bayesian graph
C	Bayesian graph with variables for modeling sample virus concentration
$l \in L$	A leaf node, corresponds to a building
$j \in J$	A non-leaf node, corresponds to a junction node in the reduced graph
W	Set of nodes (leaf/non-leaf nodes) where the sensors are placed.
W_b	The virus outbreak state of the sensing nodes W
S	Virus outbreak scenario
k	Number of sensors available to be deployed
b	Random variable to indicate whether samples from a node contain viruses
$\hat{n}$	Random variable to model the number of infected individuals in a building
n	Random variable to model the number of total virus copies shed from a building
f	Random variable to model the volume of wastewater flowing from a building
c	Random variable to model the concentration of a wastewater sample collected at a node

6.4 Simulation Experiments

We tested our approach by analyzing a subgraph of our university's wastewater network consisting of residential buildings[2]. Initially, we constructed the reduced graph representation of the wastewater network, denoted as G, by following the approach outlined in Section 6.3.1. The original wastewater network graph contained 35 vertices and 34 edges. However, we were able to reduce it to a 20 vertex and 19 edge graph using our graph reduction approach (shown in Figure 6.6). This corresponds to a 41% reduction in the number of vertices and a 44% reduction in the number of edges of the graph. Out of the 20 vertices, 12 represented leaf nodes which corresponded to buildings.

[2]Some aspects of the wastewater network have been obfuscated for security and confidentiality.

Next, we utilized the wastewater network graph G to construct two Bayesian graph networks: B to model virus presence states, and C to model wastewater concentrations. Our methodology for constructing B and C is described in Section 6.3.2 and Section 6.3.5, respectively.

We modeled the wastewater flow volume f of each building in the Bayesian graph C using truncated Gaussian distributions, which were parameterized based on the buildings' historical monthly wastewater flow rates. The Poisson distribution was used to model the number of infected individuals, and the mean of this distribution was set to be proportional to the number of students assigned to the corresponding building. Lastly, we bounded the uniform distribution over the daily number of virus copies shed in the wastewater by each infected individual between 2.4×10^6 and 4×10^{10}, based on the findings of Foladori et al. [136].

To generate hypothetical scenarios, we utilized Poisson distributions in the auxiliary graph C. We generated a total of 1000 scenarios, some of which modeled simultaneous virus outbreaks in multiple buildings. These scenarios were binarized as described in Section 6.3.5. We used these scenarios in all of our experiments, unless specified otherwise, to maintain consistency in our benchmark results.

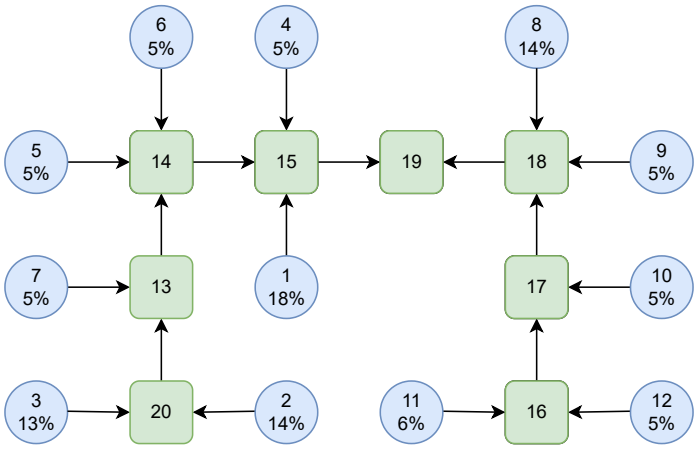


Figure 6.6: Subgraph of our university's wastewater network after reduction. The percentages in the leaf nodes (blue circles) indicate the fraction of the total network population at each node.

6.4.1 Optimization Metric Benchmark

We began by benchmarking various score functions in Equation 6.7 which is used to determine sensor placements. As discussed in Section 6.3.4, accuracy is not the most effective optimization metric for our approach. Therefore, we employed the greedy algorithm to optimize sensor placements using three other score functions: precision, recall, and F1 in addition to accuracy. We set the virus concentration in the threshold indicator function (Equation 6.8) to 4.8×10^5 and generated solutions for 6 sensor placements. Figure 6.7 displays the quality of the sensor placements obtained by optimizing with each score function. We evaluated the solution placements using all four score functions and reported them, along with the fraction of scenarios that could be covered (Equation 6.8) using the solution placements.

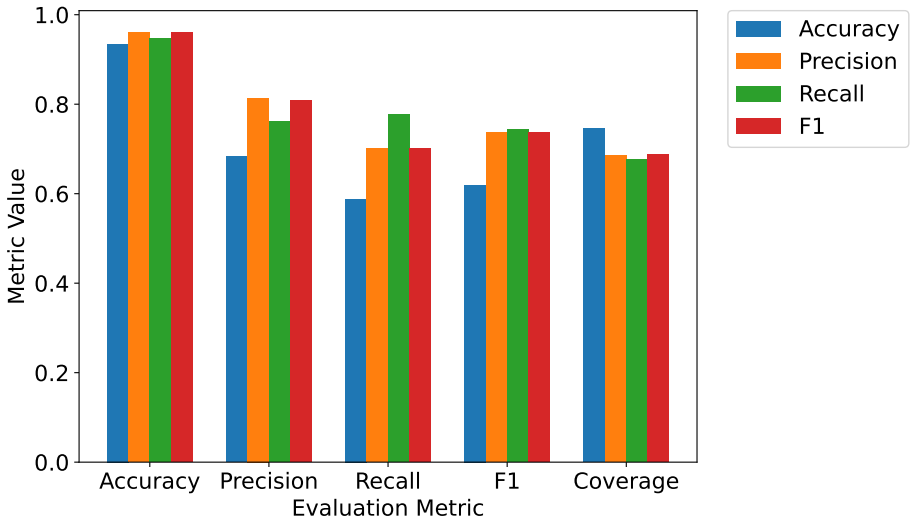


Figure 6.7: Optimization metric benchmark.

As expected, we achieved the best results for each score function when the optimization metric matched the evaluation metric. In a real-world scenario, the appropriate optimization function can be selected based on the fraction of false positive and true negative source localization predictions that can be accommodated. For the remaining experiments, we utilized only the F1 score as the optimization score function since

optimizing this score provided us with good precision and recall during evaluation, even though they were not directly optimized.

6.4.2 Optimizer Benchmark and Submodularity

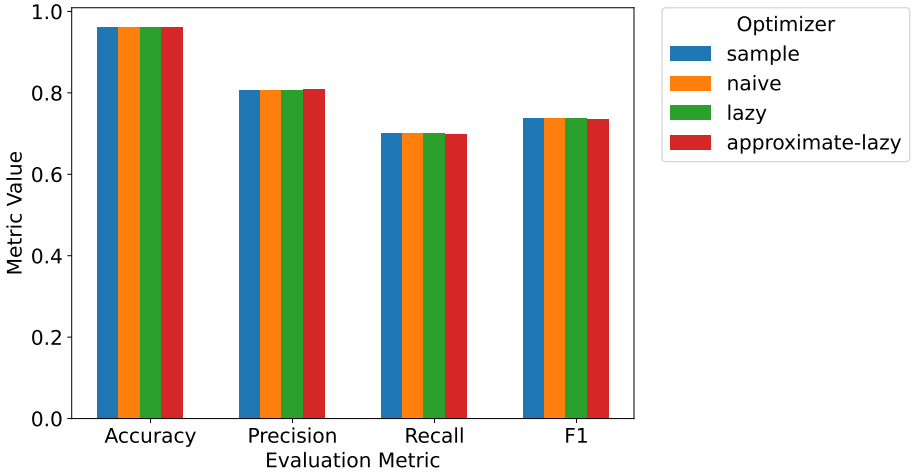


Figure 6.8: Optimizer benchmark.

In addition to using different score functions, our approach also allows for the use of different optimizers. We benchmarked four optimizers—naive [135], lazy [135], approximate-lazy [137], and sample [138]. We generated solutions for six sensor placements using a virus concentration of 4.8×10^5 in the threshold indicator function (Equation 6.8), the results are shown in Figure 6.8.

The lazy, approximate-lazy, and sampls optimizers offer faster solutions than the naive optimizer without sacrificing quality, and provide a near-optimal $(1 - 1/e)$ approximation factor guarantee. This is because they assume that the optimization objective is submodular [139]. We can observe that all of these approaches perform similarly to the naive optimizer, which is possible only if our optimization objective is submodular.

6.4.3 Weighted Sum Benchmark

In our previous experiments, we used an unweighted sum of the score function and the indicator function value in our optimization objective (Equation 6.7). However,

in a real-world scenario, it may be necessary to prioritize the score function or the scenario coverage (with the indicator function). As such, we introduce a weight term λ to take a weighted sum of the score and indicator function values during optimization:

$$\underset{W \subset V, |W| \leq k}{\arg\max} \; \mathbb{E}_S[(\lambda) score(A(W_b), S) + (1-\lambda)\mathbb{1}_{\text{thresh}}(W, S)].$$

To illustrate the impact of different λ values, we benchmarked it by generating solutions using various λ values, and the results are presented in Figure 6.9. We utilized the naive greedy optimizer [135], set the virus concentration in the threshold indicator function (Equation 6.8) at 4.8×10^5, and produced solutions for 6 sensor placements.

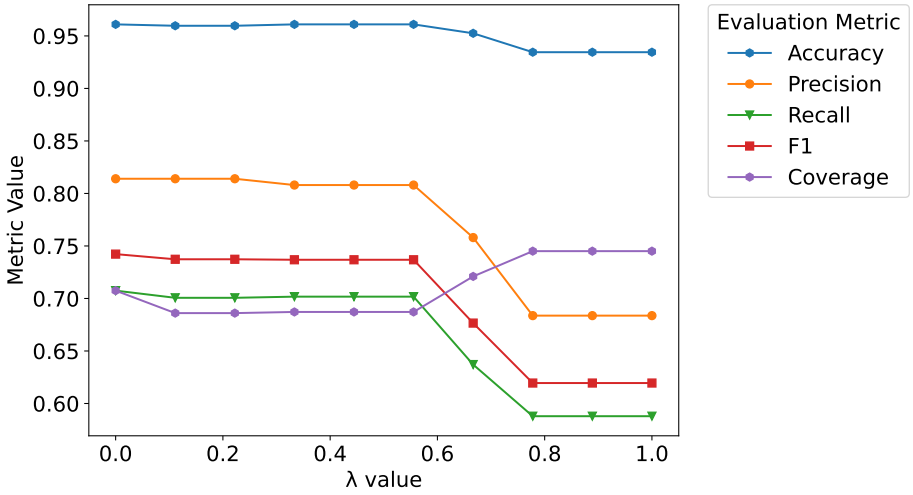


Figure 6.9: Weighted Sum Benchmark.

We observe from the plot that optimizing solely for coverage using the indicator function (left side) yields high score values, but the scenario coverage is somewhat reduced. Conversely, optimizing only for the score function on the right side of the plot improves the coverage, albeit with a decrease in the score values.

6.4.4 Concentration Threshold Benchmark

We also benchmarked our solution quality with different concentration values in the threshold indicator function (Equation 6.8). We used the naive greedy optimizer [135] and generated solutions for 6 sensor placements. Figure 6.10 shows our results.

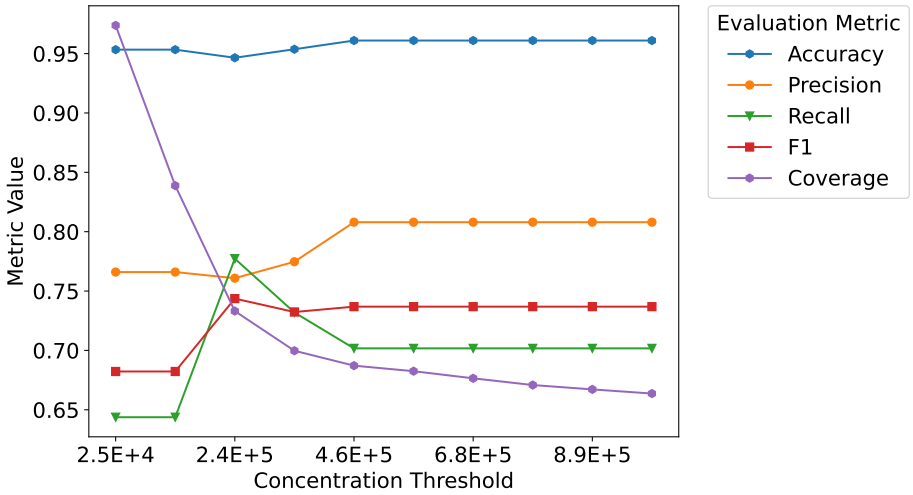
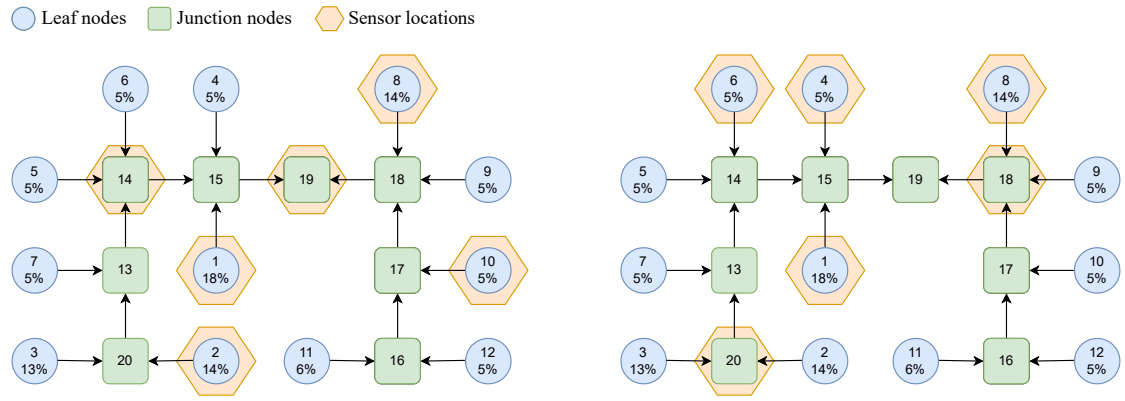


Figure 6.10: Concentration Threshold Benchmark.

As we had anticipated, using low concentration thresholds enables us to detect a greater portion of the outbreak scenarios. This is due to the fact that the sensor placements are closer to the root node, where all wastewater ultimately flows. Consequently, detecting most virus outbreaks is feasible as the low virus concentration is not an issue. However, as we increase the concentration threshold, sensors must be repositioned closer to the leaf nodes (i.e., buildings) to ensure that the wastewater samples contain high virus concentrations. As a result, we would require more sensors to cover all the buildings. Nevertheless, this has the added benefit of improved source localization as each sensor is allocated to a smaller subgraph with fewer leaf nodes.

Figure 6.11 illustrates the solution placements on our university's wastewater network graph. We observe that the solution generated without a concentration threshold (Figure 6.11(a)) provides inadequate coverage (55.28%) as the wastewater at the root node (i.e., node 19) is too diluted to produce reliable virus detection results.



(a) Without concentration threshold. Coverage: 55.28%, F1: 67.33%

(b) With concentration threshold. Coverage: 74.51%, F1: 61.95%

Figure 6.11: Solution placements for a subgraph of our university's wastewater network. (a) Solution computed without a concentration threshold. (b) Solution computed with a concentration threshold of 4.8×10^5 virus copies per liter. The percentages in the leaf nodes (blue circles) indicate the fraction of the total network population at each node. The solution placements were evaluated with a concentration threshold of 4.8×10^5 virus copies per liter.

However, the solution generated with a threshold of 4.8×10^5 virus copies per liter (Figure 6.11(b)) delivers substantially improved coverage (74.51%).

6.4.5 Detection Threshold Benchmark

In another experiment, we evaluated the impact of varying the virus detection threshold. To determine the virus outbreak state of each building, we thresholded the state probabilities (Equation 6.1), i.e., label the state as $True$ if the probability is above the threshold, even if the $False$ probability is higher than the $True$ probability. Figure 6.12 illustrates the solution quality obtained using different detection thresholds. For all evaluations in this benchmark, we used the same sensor placements acquired in Section 6.4.1 with the F1 score objective. The sensor placements consisted of 6 sensors and were optimized using the naive greedy optimizer [135].

Note that the zero threshold corresponds to using the original argmax operation (Equation 6.1) to determine the state of each building. The results demonstrate the classic inverse relation between precision and recall. Precision heavily penalizes the score if we predict a building with a $False$ virus outbreak state as $True$ (false

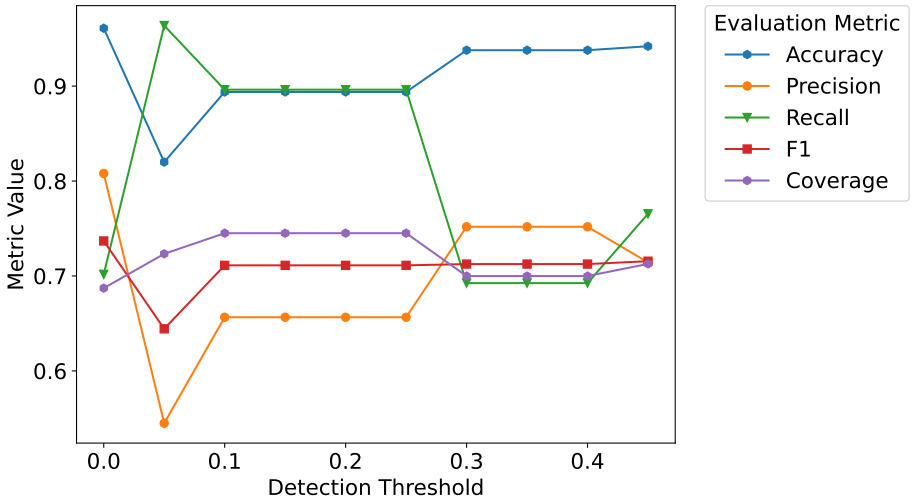


Figure 6.12: Probability Detection Threshold Benchmark.

positives). On the other hand, recall allows us to make the same prediction without being penalized, but instead, it penalizes the score if we predict a building with a $True$ virus outbreak state as $False$ (false negative). As such, we observe that increasing the detection threshold initially improves recall, as it increases the fraction of true positives at the cost of increased false positives. However, beyond a threshold of 0.25, it improves precision instead, as a larger fraction of predictions become true positives but with increased false negatives.

6.4.6 Random Graph Benchmark

The experiments we conducted previously focused on a subgraph of our university's wastewater network, allowing us to isolate and study the effects of each variable of interest. In this experiment, we aimed to test the generalizability of our approach to different wastewater networks. To do so, we generated 20 random wastewater network graphs and their corresponding outbreak scenarios, and then generated sensor placement solutions for each graph.

Our random graphs consisted of 25 nodes each. The graphs were built by iteratively linking a new node to a randomly selected existing node. Additionally, we redirected new nodes to successor nodes of the randomly selected nodes, away from the root

with a probability of 20%. We then sampled the building populations from a uniform distribution in the interval $[0, 100]$, and the wastewater flows for each leaf node from a uniform distribution in the interval $[1000, 3000]$.

For each graph, we sampled 500 virus outbreak scenarios and optimized the placement of 6 sensors using the naive greedy algorithm. The thresholded coverage objective with the F1 score (Equation 6.7) was used as the optimization objective. We set the concentration threshold to 4.8×10^5 virus copies per liter.

Figure 6.13 and Figure 6.14 (Unperturbed) show the average solution F1 score and coverage on the 20 random graphs. We compared our results to a baseline of randomly placed sensors (Random) and found that our approach produced substantial improvements in performance, comparable to those obtained on our university's wastewater network. Note that the random graphs have varying connectivity and different ratios of sensor placements to graph size when compared to our university's wastewater network.

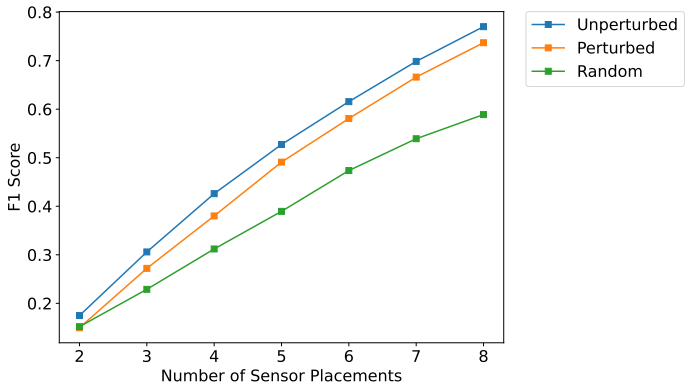


Figure 6.13: F1 scores on the Random Graph Benchmark.

To test the limits of our approach further, we perturbed the virus outbreak probability of each building in the random graphs and generated a new set of scenarios for each random graph. We perturbed the virus outbreak probabilities by adding uniform-distributed noise to the populations of each building in the random graphs. We evaluated the previously obtained sensor placement solutions that were optimized

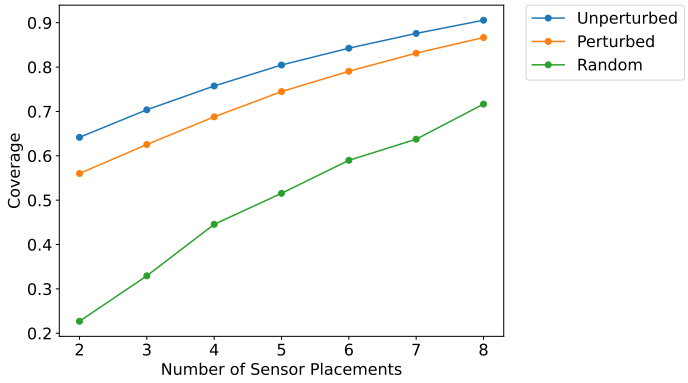


Figure 6.14: Coverage rates on the Random Graph Benchmark.

on the unperturbed virus outbreak probability-based scenarios using the new scenarios. This experiment allowed us to test our approach in real-world scenarios where the available virus outbreak probability or the whole model itself may be inconsistent with the actual virus outbreak characteristics. The results, shown in Figure 6.13 and Figure 6.14 (Perturbed), demonstrate that our approach is robust to modeling inconsistencies.

6.5 Related Work

There has been an active response to COVID-19 in automation and robotics, using robots for disinfection, monitoring patients, making deliveries, lab automation, and for telemedicine. See [140, 141] for comprehensive surveys. However, existing work has not addressed the problem considered in this chapter.

Our problem is similar to that of Kempe et al. [142], who studied influence maximization in social networks. The authors considered social influence networks and developed an approach to find a subset of nodes that, when influenced to take a specified action, would result in the maximal number of nodes in the entire network taking the same action. We are also interested in extracting maximal information about the graph by monitoring a small number of nodes. However, we also need to identify the sources of information.

Berry et al. [143] developed a mixed-integer programming problem formulation

for sensor placement in water distribution networks. The method determined sensor placement locations that would detect any contaminants in water in the shortest time possible. Leskovec et al. [133] also studied the sensor placement problem in water distribution networks and developed a submodular objective. Their approach could efficiently optimize the objective to find a solution with provable approximation guarantees. However, both approaches did not address source localization.

Spinelli et al. [144] developed an approach for source localization in contact networks to curb the spread of epidemics. The method considered disease transmission times between patients to identify patient zero. However, the approach is limited to single-source scenarios.

Jiang et al. [145] proposed a Bayesian approach to disease outbreak detection and prediction by examining particular variables of interest that could indicate an outbreak in a given community. However the method did not consider identifying the outbreak's source and was limited to predicting only the presence of an epidemic and its scale.

Jiang et al. [146] developed a Bayesian spatial scan statistic that considered indicator variables to identify the presence and location of an outbreak. But the localization approach assumed a rectilinear partitioning of the region of interest and access to data from every sub-region.

The source localization problem also appears frequently in other domains such as underwater localization [147], wireless network user localization [148], and EEG device signal localization [149]. However most solutions to such problems develop application specific metrics and optimization objectives.

Peccia et al. [126] studied the statistics of the COVID-19 virus in wastewater samples to identify general trends of community infection rates. Gibas et al. [132] examined the feasibility of sampling wastewater in a university to identify COVID-19 outbreaks and demonstrated the ability to detect single asymptomatic individuals in

a dormitory. However, these papers did not consider sensor placement optimization.

Recently sensor placement approaches specifically for wastewater monitoring have been presented [150, 151, 152, 153, 154]. These approaches focused on sensor placement to maximize the population coverage while minimizing each sensor's coverage overlap, which was achieved using discrete integer programs. Nourinejad et al. [150] and Calle et al. [151] were among the first to develop sensor placement approaches for wastewater-based epidemiology. They presented Bayesian approaches to iteratively locate virus hotspots and separately leveraged the graph structure of wastewater networks to optimize the sensor placement locations using discrete optimization.

Despite these advances, researchers in wastewater-based epidemiology have yet to fully utilize Bayesian approaches, especially *graph* Bayesian methods [134], which can leverage graph structures to model causal relationships in the problem. By using efficient graph Bayesian methods such as message-passing techniques, we can reduce computation costs and develop superior source localization methods. Additionally, we have demonstrated the potential of optimizing sensor placement for source localization accuracy directly, while also incorporating virus infection rates and wastewater sample concentration requirements into our models.

6.6 Conclusion

We presented the sensor placement for source localization problem in wastewater networks, and developed an approach that leverages graph Bayesian learning and discrete optimization to address the problem.

We first presented an approach to reduce the size of wastewater network graphs, thereby making relatively large problems computationally feasible. We then showed how one can map any network graph to a Bayesian graph, which we can use to localize sources of information, i.e., buildings experiencing a virus outbreak in our case. We also developed optimization objectives that we can use to efficiently find ideal sensor placements for source localization, even when there are multiple pathogen sources

and constraints such as wastewater concentration requirements.

Our simulation experiments demonstrated the quality of our solution sensor placements and the accuracy of our source localization approach in a case study on our university's wastewater network. We also benchmarked different discrete optimization methods and score functions, and showed that our optimization objective can be efficiently optimized. We then established the accuracy of our approach on random graphs. In addition, our experiments established the robustness of our approach to inaccurate virus outbreak models.

In a real-world scenario, our graph sensor placement approach coupled with virus outbreak models and information about the wastewater network structure can determine ideal wastewater sampling locations. Our source localization approach can quickly localize the source of virus outbreaks from regularly collected wastewater sample test results. Additionally, our graph reduction and source localization approaches can be used with existing wastewater network sensor placements approaches.

Our experiments have demonstrated that our optimization objective is submodular on our university's wastewater network graph. We conjecture that this property holds for any wastewater network graph, and we plan to prove this theoretically in our future work.

Furthermore, to validate our approach using real-world data, we require regular virus concentration measurements and the number of infected individuals for each building in the monitored wastewater network. We aim to collect such data and present additional validation of our approach in future work.

CHAPTER 7: CONCLUSION

This thesis addressed the sensor placement problem and the closely related informative path planning problem. First, fundamental concepts related to these issues were detailed. Second, the sensor placement problem in discrete and continuous spaces was tackled by leveraging the inherent properties of sparse Gaussian processes. The method was then generalized to address non-point and integrated field-of-view sensors. This, in turn, was leveraged to tackle the informative path planning problem. Indeed, we further generalized our method to efficiently handle multiple robots, spatio-temporal data fields, incorporate past data into the path planning phase, and efficiently learn the model parameters. Then the offline IPP method was further generalized to effectively handle online IPP using streaming sparse Gaussian processes. Moreover, a K-nearest neighbour based environment partitioning scheme was presented to enable decentralized multi-robot IPP with minimal communication requirements.

Next, the thesis addressed the informative path planning problem in graphs for source localization. In particular, we considered the localization of methane gas leaks in oil fields using gas concentration data collected from vehicles traversing the road network. The proposed method leveraged the properties of Gaussian distributions to significantly reduce the computation time for computing mutual information, by orders of magnitude compared to the baseline. Moreover, we presented an efficient algorithm for finding informative paths that maximize mutual information.

Finally, the thesis addressed the sensor placement problem in graphs for source localization. Indeed, we focused on localizing virus sources in wastewater networks. The problem was approached by presenting an efficient algorithm to model the wastewater

network as a Bayesian network. The Bayesian network was then utilized to formulate discrete optimization objectives that could be optimized to determine the optimal sensor placements. Moreover, the Bayesian networks were employed to efficiently locate the virus sources using Bayesian inference.

The methods developed in this thesis can be applied to a broader range of problems including the following:

- Informative path and trajectory planning: Our IPP approaches focus on path planning, where we decide where the robot should go. In the trajectory planning problem, we must also determine how the robot follows the path. This includes determining the robot's velocity and acceleration, and modeling any non-holonomic constraints on the robot.

- Decentralized IPP with collective hyperparameter learning and periodic environment partitioning: Our decentralized IPP approach partitions the environment only at the beginning of the online multi-robot IPP phase and does not require any inter-robot communication. However, when considering a dynamic environment, developing methods for the robots to synchronize their hyperparameters and update their monitoring regions could result in better environment monitoring.

- IPP for 3D surface inspection: Our IPP experiments focused on environment monitoring problems. However, our approaches can also leverage Riemannian kernel functions to model IPP on 3D surfaces, such as aircraft fuselages, for inspection tasks.

- Deriving approximation proofs for our SGP-based IPP approaches: Future work can include deriving approximation proofs for the environment coverage fraction that can be achieved using our IPP approaches.

REFERENCES

[1] O. Hamelijnck, W. J. Wilkinson, N. A. Loppi, A. Solin, and T. Damoulas, "Spatio-Temporal Variational Gaussian Processes," in *Advances in Neural Information Processing Systems* (A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), 2021.

[2] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, USA: MIT Press, 2005.

[3] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[4] M. Khan and W. Lin, "Conjugate-Computation Variational Inference : Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Models," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, pp. 878–887, PMLR, 20–22 Apr 2017.

[5] W. J. Wilkinson, S. Särkkä, and A. Solin, "Bayes-Newton Methods for Approximate Bayesian Inference with PSD Guarantees," *Journal of Machine Learning Research*, vol. 24, no. 83, pp. 1–50, 2023.

[6] M. E. Khan and H. Rue, "The bayesian learning rule," *Journal of Machine Learning Research*, vol. 24, no. 281, pp. 1–46, 2023.

[7] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using Pseudo-inputs," in *Advances in Neural Information Processing Systems* (Y. Weiss, B. Schölkopf, and J. Platt, eds.), vol. 18, MIT Press, 2006.

[8] M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics* (D. van Dyk and M. Welling, eds.), (Florida, USA), pp. 567–574, PMLR, 2009.

[9] T. N. Hoang, Q. M. Hoang, and B. K. H. Low, "A Unifying Framework of Anytime Sparse Gaussian Process Regression Models with Stochastic Variational Inference for Big Data," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37, (Lille, France), pp. 569–578, PMLR, 2015.

[10] T. D. Bui, J. Yan, and R. E. Turner, "A Unifying Framework for Gaussian Process Pseudo-Point Approximations Using Power Expectation Propagation," *Journal of Machine Learning Research*, vol. 18, no. 104, pp. 1–72, 2017.

[11] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding Probabilistic Sparse Gaussian Process Approximations," in *Advances in Neural Information Processing Systems*, (Red Hook, NY, USA), p. 1533–1541, 2016.

[12] D. Burt, C. E. Rasmussen, and M. Van Der Wilk, "Rates of Convergence for Sparse Variational Gaussian Process Regression," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97, pp. 862–871, PMLR, Jun 2019.

[13] A. Krause, A. Singh, and C. Guestrin, "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies," *Journal of Machine Learning Research*, vol. 9, no. 8, pp. 235–284, 2008.

[14] K.-C. Ma, L. Liu, and G. S. Sukhatme, "Informative Planning and Online Learning with Sparse Gaussian Processes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4292–4298, 2017.

[15] V. Suryan and P. Tokekar, "Learning a Spatial Field in Minimum Time With a Team of Robots," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1562–1576, 2020.

[16] J. Whitman, H. Maske, H. A. Kingravi, and G. Chowdhary, "Evolving Gaussian Processes and Kernel Observers for Learning and Control in Spatiotemporally Varying Domains: With Applications in Agriculture, Weather Monitoring, and Fluid Dynamics," *IEEE Control Systems*, vol. 41, pp. 30–69, 2021.

[17] K. Jakkala and S. Akella, "Probabilistic Gas Leak Rate Estimation Using Submodular Function Maximization With Routing Constraints," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5230–5237, 2022.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Adaptive Computation and Machine Learning series, MIT Press, 2016.

[19] H. Zhu, J. J. Chung, N. R. Lawrance, R. Siegwart, and J. Alonso-Mora, "Online Informative Path Planning for Active Information Gathering of a 3D Surface," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1488–1494, 2021.

[20] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively Learning Gaussian Process Dynamics," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control* (A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, eds.), vol. 120 of *Proceedings of Machine Learning Research*, pp. 5–15, PMLR, 10–11 Jun 2020.

[21] T. Husain and W. F. Caselton, "Hydrologic Network Design Methods and Shannon's Information Theory," *IFAC Proceedings Volumes*, vol. 13, no. 3, pp. 259–267, 1980. IFAC Symposium on Water and Related Land Resource Systems, Cleveland, OH, USA, May 1980.

[22] M. C. Shewry and H. P. Wynn, "Maximum entropy sampling," *Journal of Applied Statistics*, vol. 14, no. 2, pp. 165–170, 1987.

[23] S. Wu and J. V. Zidek, "An entropy-based analysis of data from selected NADP/NTN network sites for 1983–1986," *Atmospheric Environment. Part A. General Topics*, vol. 26, no. 11, pp. 2089–2103, 1992.

[24] J. Quinonero-Candela, C. E. Rasmussen, and C. K. I. Williams, "Approximation Methods for Gaussian Process Regression," in *Large-Scale Kernel Machines*, pp. 203–223, MIT Press, 2007.

[25] C. Bigoni, Z. Zhang, and J. S. Hesthaven, "Systematic sensor placement for structural anomaly detection in the absence of damaged states," *Computer Methods in Applied Mechanics and Engineering*, vol. 371, p. 113315, 2020.

[26] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai, "Deploying Wireless Sensors to Achieve Both Coverage and Connectivity," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, (New York, NY, USA), p. 131–142, 2006.

[27] D. Ramsden, *Optimization approaches to sensor placement problems*. PhD thesis, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, August 2009.

[28] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Berlin: Springer-Verlag, third ed., 2008.

[29] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[30] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE International Conference on Robotics and Automation*, pp. 4982–4989, 2010.

[31] A. Sadeghi, A. B. Asghar, and S. L. Smith, "Distributed multi-robot coverage control of non-convex environments with guarantees," *IEEE Transactions on Control of Network Systems*, pp. 1–12, 2022.

[32] M. Schwager, M. P. Vitus, S. Powers, D. Rus, and C. J. Tomlin, "Robust adaptive coverage control for robotic sensor networks," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 3, pp. 462–476, 2017.

[33] T. Salam and M. A. Hsieh, "Adaptive sampling and reduced-order modeling of dynamic processes by robot teams," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 477–484, 2019.

[34] G. Tajnafoi, R. Arcucci, L. Mottet, C. Vouriot, M. Molina-Solana, C. Pain, and Y.-K. Guo, "Variational gaussian process for optimal sensor placement," *Applications of Mathematics*, vol. 66, pp. 287–317, Apr 2021.

[35] X. Lin, A. Chowdhury, X. Wang, and G. Terejanu, "Approximate computational approaches for Bayesian sensor placement in high dimensions," *Information Fusion*, vol. 46, pp. 193–205, 2019.

[36] G. Hitz, E. Galceran, M.-E. Garneau, F. Pomerleau, and R. Siegwart, "Adaptive Continuous-Space Informative Path Planning for Online Environmental Monitoring," *Journal of Field Robotics*, vol. 34, no. 8, pp. 1427–1449, 2017.

[37] G. Francis, L. Ott, R. Marchant, and F. Ramos, "Occupancy map building through Bayesian exploration," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 769–792, 2019.

[38] R. Semaan, "Optimal sensor placement using machine learning," *Computers & Fluids*, vol. 159, pp. 167–176, 2017.

[39] Z. Wang, H.-X. Li, and C. Chen, "Reinforcement learning-based optimal sensor placement for spatiotemporal modeling," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2861–2871, 2020.

[40] B. Li, H. Liu, and R. Wang, "Efficient sensor placement for signal reconstruction based on recursive methods," *IEEE Transactions on Signal Processing*, vol. 69, pp. 1885–1898, 2021.

[41] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, p. 859–877, Apr. 2017.

[42] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian Processes for Big Data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, (Arlington, Virginia, USA), p. 282–290, AUAI Press, 2013.

[43] D. R. Burt, C. E. Rasmussen, and M. van der Wilk, "Convergence of Sparse Variational Inference in Gaussian Processes Regression," *Journal of Machine Learning Research*, vol. 21, no. 131, pp. 1–63, 2020.

[44] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems: revised reprint*. Philadelphia, USA: SIAM, 2012.

[45] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel lab data," *Online dataset*, 2004.

[46] C. Bretherton, M. Widmann, V. Dymnikov, J. Wallace, and I. Bladé, "The effective number of spatial degrees of freedom of a time-varying field," *Journal of Climate*, vol. 12, no. 7, pp. 1990–2009, 1999.

[47] C. Hain, "NASA SPoRT-LiS Soil Moisture Products," 2013.

[48] A. F. Shchepetkin and J. C. McWilliams, "The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model," *Ocean Modelling*, vol. 9, no. 4, pp. 347–404, 2005.

[49] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (Poster)*, 2015.

[50] M. van der Wilk, V. Dutordoir, S. John, A. Artemev, V. Adam, and J. Hensman, "A Framework for Interdomain and Multioutput Gaussian Processes," *ArXiv*, 2020.

[51] J. Schreiber, J. Bilmes, and W. S. Noble, "apricot: Submodular selection for data summarization in Python," *Journal of Machine Learning Research*, vol. 21, no. 161, pp. 1–6, 2020.

[52] C. C. C. Service, "Ozone monthly gridded data from 1970 to present derived from satellite observations," 2018.

[53] M. McKay, R. Beckman, and W. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code," *Technometrics*, vol. 21, pp. 239–245, 05 1979.

[54] S. Remes, M. Heinonen, and S. Kaski, "Neural non-stationary spectral kernel," *ArXiv*, 2018.

[55] J. Binney, A. Krause, and G. S. Sukhatme, "Informative path planning for an autonomous underwater vehicle," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4791–4796, 2010.

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

[57] J. Binney, A. Krause, and G. S. Sukhatme, "Optimizing waypoints for monitoring spatiotemporal phenomena," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 873–888, 2013.

[58] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent Ocean Monitoring with Underwater Gliders: Adapting Sampling Resolution," *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[59] J. Rückin, L. Jin, F. Magistri, C. Stachniss, and M. Popović, "Informative Path Planning for Active Learning in Aerial Semantic Mapping," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11932–11939, 2022.

[60] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[61] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Int. Res.*, vol. 34, p. 707–755, Apr. 2009.

[62] G. Hollinger and S. Singh, "Proofs and experiments in scalable, near-optimal search by multiple robots," in *Robotics: Science and Systems IV*, pp. 206–213, 2009.

[63] C. Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pp. 245–253, 2005.

[64] A. Krause and C. Guestrin, "Submodularity and its applications in optimized information gathering," *ACM Trans. Intell. Syst. Technol.*, vol. 2, July 2011.

[65] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Int. Res.*, vol. 34, p. 707–755, Apr. 2009.

[66] L. Bottarelli, M. Bicego, J. Blum, and A. Farinelli, "Orienteering-based informative path planning for environmental monitoring," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 46 – 58, 2019.

[67] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic Exploration of Distributed Information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2016.

[68] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto, "An informative path planning framework for UAV-based terrain monitoring," *Autonomous Robots*, vol. 44, pp. 889–911, Jul 2020.

[69] K. C. T. Vivaldini, T. H. Martinelli, V. C. Guizilini, J. R. Souza, M. D. Oliveira, F. T. Ramos, and D. F. Wolf, "UAV route planning for active disease classification," *Autonomous Robots*, vol. 43, pp. 1137–1153, Jun 2019.

[70] R. Mishra, M. Chitre, and S. Swarup, "Online Informative Path Planning Using Sparse Gaussian Processes," in *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, pp. 1–5, 2018.

[71] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 342–349, 2014.

[72] S. Agarwal and S. Akella, "The Correlated Arc Orienteering Problem," in *Algorithmic Foundations of Robotics XV* (S. M. LaValle, J. M. O'Kane, M. Otte, D. Sadigh, and P. Tokekar, eds.), (Cambridge, Massachusetts, USA), pp. 402–418, Springer International Publishing, 2023.

[73] J. Rückin, L. Jin, and M. Popović, "Adaptive Informative Path Planning Using Deep Reinforcement Learning for UAV-based Active Sensing," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 4473–4479, 2022.

[74] K. Jakkala and S. Akella, "Efficient Sensor Placement from Regression with Sparse Gaussian Processes in Continuous and Discrete Spaces." Manuscript submitted for publication., 2023.

[75] L. Perron and V. Furnon, "OR-Tools."

[76] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Philadelphia, USA: Society for Industrial and Applied Mathematics, 2014.

[77] K. Jakkala and S. Akella, "Multi-robot informative path planning from regression with sparse Gaussian processes," in *IEEE International Conference on Robotics and Automation, ICRA*, (Yokohama, Japan), May 2024. `https://arxiv.org/pdf/2309.07050.pdf`.

[78] R. Murray-Smith and B. A. Pearlmutter, "Transformations of Gaussian Process Priors," in *Deterministic and Statistical Methods in Machine Learning* (J. Winkler, M. Niranjan, and N. Lawrence, eds.), (Berlin, Heidelberg), pp. 110–123, Springer, 2005.

[79] K. Longi, C. Rajani, T. Sillanpää, J. Mäkinen, T. Rauhala, A. Salmi, E. Haeggström, and A. Klami, "Sensor Placement for Spatial Gaussian Processes with Integral Observations," in *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)* (J. Peters and D. Sontag, eds.), vol. 124 of *Proceedings of Machine Learning Research*, pp. 1009–1018, PMLR, 03–06 Aug 2020.

[80] G. L. Zeng, *Image Reconstruction: Applications in Medical Sciences*. De Gruyter, 2017.

[81] M. Jun, G. Cheng, W. Yixin, A. Xingle, G. Jiantao, Y. Ziqi, Z. Minqing, L. Xin, D. Xueyuan, C. Shucheng, W. Hao, M. Sen, Y. Xiaoyu, N. Ziwei, L. Chen, T. Lu, Z. Yuntao, Z. Qiongjie, D. Guoqiang, and H. Jian, "COVID-19 CT Lung and Infection Segmentation Dataset," Apr. 2020.

[82] W. van Aarle, W. J. Palenstijn, J. De Beenhouwer, T. Altantzis, S. Bals, K. J. Batenburg, and J. Sijbers, "The astra toolbox: A platform for advanced algorithm development in electron tomography," *Ultramicroscopy*, vol. 157, pp. 35–47, 2015.

[83] P. Ghassemi, M. Balazon, and S. Chowdhury, "A penalized batch-Bayesian approach to informative path planning for decentralized swarm robotic search," *Autonomous Robots*, vol. 46, pp. 725–747, Aug 2022.

[84] A. A. R. Newaz, M. Alsayegh, T. Alam, and L. Bobadilla, "Decentralized Multi-Robot Information Gathering From Unknown Spatial Fields," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 3070–3077, 2023.

[85] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.

[86] G. E. Berget, J. Eidsvik, M. O. Alver, and T. A. Johansen, "Dynamic stochastic modeling for adaptive sampling of environmental variables using an auv," *Autonomous Robots*, vol. 47, pp. 483–502, Apr 2023.

[87] B. Moon, S. Chatterjee, and S. Scherer, "TIGRIS: An Informed Sampling-based Algorithm for Informative Path Planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5760–5766, 2022.

[88] Y. Cao, Y. Wang, A. Vashisth, H. Fan, and G. A. Sartoretti, "CAtNIPP: Context-Aware Attention-based Network for Informative Path Planning," in *6th Annual Conference on Robot Learning*, 2022.

[89] T. D. Bui, C. Nguyen, and R. E. Turner, "Streaming sparse Gaussian process approximations," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[90] "Digital coast." coast.noaa.gov. `https://coast.noaa.gov/digitalcoast`.

[91] U.S. Environmental Protection Agency, "Inventory of US greenhouse gas emissions and sinks," Tech. Rep. EPA 430-R-21-005, U.S. Environmental Protection Agency, 2021.

[92] K. Hayhoe, H. S. Kheshgi, A. K. Jain, and D. J. Wuebbles, "Substitution of natural gas for coal: climatic effects of utility sector emissions," *Climatic Change*, vol. 54, no. 1, pp. 107–139, 2002.

[93] D. Farquharson, P. Jaramillo, G. Schivley, K. Klima, D. Carlson, and C. Samaras, "Beyond global warming potential: A comparative application of climate impact metrics for the life cycle assessment of coal and natural gas based electricity," *Journal of Industrial Ecology*, vol. 21, no. 4, pp. 857–873, 2017.

[94] R. A. Alvarez, D. Zavala-Araiza, D. R. Lyon, D. T. Allen, Z. R. Barkley, A. R. Brandt, K. J. Davis, S. C. Herndon, D. J. Jacob, A. Karion, E. A. Kort, B. K. Lamb, T. Lauvaux, J. D. Maasakkers, A. J. Marchese, M. Omara, S. W. Pacala, J. Peischl, A. L. Robinson, P. B. Shepson, C. Sweeney, A. Townsend-Small, S. C. Wofsy, and S. P. Hamburg, "Assessment of methane emissions from the U.S. oil and gas supply chain," *Science*, vol. 361, no. 6398, pp. 186–188, 2018.

[95] J. D. Albertson, T. Harvey, G. Foderaro, P. Zhu, X. Zhou, S. Ferrari, M. S. Amin, M. Modrak, H. Brantley, and E. D. Thoma, "A mobile sensing approach for regional surveillance of fugitive methane emissions in oil and gas production," *Environmental Science & Technology*, vol. 50, no. 5, pp. 2487–2497, 2016.

[96] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, p. 819–825, 2016.

[97] S. Pandey, R. Gautam, S. Houweling, H. D. van der Gon, P. Sadavarte, T. Borsdorff, O. Hasekamp, J. Landgraf, P. Tol, T. van Kempen, R. Hoogeveen, R. van Hees, S. P. Hamburg, J. D. Maasakkers, and I. Aben, "Satellite observations reveal extreme methane leakage from a natural gas well blowout," *Proceedings of the National Academy of Sciences*, vol. 116, no. 52, pp. 26376–26381, 2019.

[98] D. Allen, "Project Astra," 2020.

[99] "Drilling productivity report," 2021.

[100] B. Travis, M. Dubey, and J. Sauer, "Neural networks to locate and quantify fugitive natural gas leaks for a MIR detection system," *Atmospheric Environment: X*, vol. 8, pp. 100092–100104, 2020.

[101] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, *An Introduction to Statistical Modeling of Extreme Values*, vol. 208. Springer, 2001.

[102] A. Corberan and G. Laporte, eds., *Arc Routing: Problems, Methods, and Applications*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2014.

[103] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[104] A. Krause and C. Guestrin, "Submodularity and its applications in optimized information gathering," *ACM Trans. Intell. Syst. Technol.*, vol. 2, July 2011.

[105] C. Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pp. 245–253, 2005.

[106] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *J. Artif. Int. Res.*, vol. 34, p. 707–755, Apr. 2009.

[107] R. Iyer and J. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, p. 2436–2444, 2013.

[108] P. P. Neumann, V. H. Bennetts, A. J. Lilienthal, M. Bartholmai, and J. H. Schiller, "Gas source localization with a micro-drone using bio-inspired and particle filter-based algorithms," *Advanced Robotics*, vol. 27, no. 9, pp. 725–738, 2013.

[109] C. Stachniss, C. Plagemann, and A. J. Lilienthal, "Learning gas distribution models using sparse Gaussian process mixtures," *Autonomous Robots*, vol. 26, p. 187–202, Apr. 2009.

[110] M. A. Arain, V. H. Bennetts, E. Schaffernicht, and A. J. Lilienthal, "Sniffing out fugitive methane emissions: Autonomous remote gas inspection with a mobile robot," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 782–814, 2021.

[111] G. N. Frederickson, "Approximation algorithms for some postman problems," *Journal of the ACM (JACM)*, vol. 26, no. 3, pp. 538–554, 1979.

[112] T. A. Foster-Wittig, E. D. Thoma, and J. D. Albertson, "Estimation of point source fugitive emission rates from a single sensor time series: A conditionally-sampled Gaussian plume reconstruction," *Atmospheric Environment*, vol. 115, pp. 101–109, 2015.

[113] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, Ltd, 1991.

[114] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 3, pp. 672–689, 2009.

[115] H. L. Brantley, E. D. Thoma, W. C. Squier, B. B. Guven, and D. Lyon, "Assessment of methane emissions from oil and gas production pads using mobile measurements," *Environmental Science & Technology*, vol. 48, no. 24, pp. 14508–14515, 2014.

[116] E. Thoma and B. Squier, "OTM 33 and OTM 33A Geospatial Measurement of Air Pollution-Remote Emissions Quantification-Direct Assessment (GMAP-REQ-DA)," *U.S. Environmental Protection Agency*, 2014.

[117] "2016 U.S. Oil & Gas Activity," Sep 2017.

[118] L. W. Beineke, O. R. Oellermann, and R. E. Pippert, "The average connectivity of a graph," *Discrete Mathematics*, vol. 252, no. 1, pp. 31–45, 2002.

[119] S. L. Priyadarsini, M. Suresh, and D. Huisingh, "What can we learn from previous pandemics to reduce the frequency of emerging infectious diseases like COVID-19?," *Global Transitions*, vol. 2, pp. 202–220, 2020.

[120] TED-Ed, "Will there be another pandemic in your lifetime?," Nov. 2022.

[121] P. M. Choi, B. J. Tscharke, E. Donner, J. W. O'Brien, S. C. Grant, S. L. Kaserzon, R. Mackie, E. O'Malley, N. D. Crosbie, K. V. Thomas, and J. F. Mueller, "Wastewater-based epidemiology biomarkers: Past, present and future," *Trends in Analytical Chemistry*, vol. 105, pp. 453–469, 2018.

[122] N. Sims and B. Kasprzyk-Hordern, "Future perspectives of wastewater-based epidemiology: Monitoring infectious disease spread and resistance to the community level," *Environment International*, vol. 139, 2020.

[123] J. Mac Mahon, A. J. Criado Monleon, L. W. Gill, J. J. O'Sullivan, and W. G. Meijer, "Wastewater-based epidemiology (WBE) for SARS-CoV-2 - A review focussing on the significance of the sewer network using a Dublin city catchment case study," *Water Science & Technology*, vol. 86, pp. 1402–1425, Sept. 2022.

[124] B. Hughes, D. Duong, B. J. White, K. R. Wigginton, E. M. G. Chan, M. K. Wolfe, and A. B. Boehm, "Respiratory syncytial virus (RSV) RNA in wastewater settled solids reflects RSV clinical positivity rates," *Environmental Science & Technology Letters*, vol. 9, no. 2, pp. 173–178, 2022.

[125] T. Luo, Z. Cao, Y. Wang, D. Zeng, and Q. Zhang, "Role of asymptomatic COVID-19 cases in viral transmission: Findings from a hierarchical community contact network model," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 576–585, 2022.

[126] J. Peccia, A. Zulli, D. E. Brackney, N. D. Grubaugh, E. H. Kaplan, A. Casanovas-Massana, A. I. Ko, A. A. Malik, D. Wang, M. Wang, J. L. Warren, D. M. Weinberger, W. Arnold, and S. B. Omer, "Measurement of SARS-CoV-2 RNA in wastewater tracks community infection dynamics," *Nature Biotechnology*, vol. 38, pp. 1164–1167, Oct 2020.

[127] D. A. Larsen and K. R. Wigginton, "Tracking COVID-19 with Wastewater," *Nature Biotechnology*, vol. 38, pp. 1151–1153, Oct 2020.

[128] S. Karthikeyan, A. Nguyen, D. McDonald, Y. Zong, N. Ronquillo, J. Ren, J. Zou, S. Farmer, G. Humphrey, D. Henderson, T. Javidi, K. Messer, C. Anderson, R. Schooley, N. K. Martin, and R. Knight, "Rapid, Large-Scale Wastewater Surveillance and Automated Reporting System Enable Early Detection of Nearly 85% of COVID-19 Cases on a University Campus," *mSystems*, vol. 6, no. 4, pp. e00793–21, 2021.

[129] Massachusetts Institute of Technology (MIT), "Testing wastewater to help detect Covid-19," Oct. 2020.

[130] V. Kisand, P. Laas, K. Palmik-Das, K. Panksep, H. Tammert, L. Albreht, H. Allemann, L. Liepkalns, K. Vooro, C. Ritz, V. Hauryliuk, and T. Tenson, "Prediction of COVID-19 positive cases, a nation-wide SARS-CoV-2 wastewater-based epidemiology study," *Water Research*, vol. 231, p. 119617, 2023.

[131] M. Wolken, T. Sun, C. McCall, R. Schneider, K. Caton, C. Hundley, L. Hopkins, K. Ensor, K. Domakonda, P. Kalvapalle, D. Persse, S. Williams, and L. B. Stadler, "Wastewater surveillance of SARS-CoV-2 and influenza in preK-12 schools shows school, community, and citywide infections," *Water Research*, vol. 231, p. 119648, 2023.

[132] C. Gibas, K. Lambirth, N. Mittal, M. A. I. Juel, V. B. Barua, L. Roppolo Brazell, K. Hinton, J. Lontai, N. Stark, I. Young, C. Quach, M. Russ, J. Kauer, B. Nicolosi, D. Chen, S. Akella, W. Tang, J. Schlueter, and M. Munir, "Implementing Building-Level SARS-CoV-2 Wastewater Surveillance on a University Campus," *Science of The Total Environment*, vol. 782, Mar. 2021.

[133] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, (New York, NY, USA), p. 420–429, Association for Computing Machinery, 2007.

[134] D. Barber, *Bayesian Reasoning and Machine Learning*. Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.

[135] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization Techniques* (J. Stoer, ed.), (Berlin, Heidelberg), pp. 234–243, Springer, 1978.

[136] P. Foladori, F. Cutrupi, N. Segata, S. Manara, F. Pinto, F. Malpei, L. Bruni, and G. La Rosa, "SARS-CoV-2 from faeces to wastewater treatment: What do we know? A review," *The Science of the Total Environment*, vol. 743, pp. 823–836, Nov 2020.

[137] K. Wei, R. Iyer, and J. Bilmes, "Fast Multi-Stage Submodular Maximization," in *Proceedings of the 31st International Conference on Machine Learning - Volume 32*, ICML'14, p. 1494–1502, JMLR, 2014.

[138] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause, "Lazier than Lazy Greedy," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, p. 1812–1818, AAAI, 2015.

[139] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, vol. 14, pp. 265–294, Dec 1978.

[140] Y. Shen, D. Guo, F. Long, L. A. Mateos, H. Ding, Z. Xiu, R. B. Hellman, A. King, S. Chen, C. Zhang, and H. Tan, "Robots under COVID-19 pandemic: A comprehensive survey," *IEEE Access*, vol. 9, pp. 1590–1615, 2021.

[141] X. V. Wang and L. Wang, "A literature survey of the robotic technologies during the COVID-19 pandemic," *Journal of Manufacturing Systems*, vol. 60, pp. 823–836, 2021.

[142] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the Spread of Influence through a Social Network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, p. 137–146, Association for Computing Machinery, 2003.

[143] J. Berry, W. E. Hart, C. A. Phillips, J. G. Uber, and J.-P. Watson, "Sensor Placement in Municipal Water Networks with Temporal Integer Programming Models," *Journal of Water Resources Planning and Management*, vol. 132, no. 4, pp. 218–224, 2006.

[144] B. Spinelli, L. E. Celis, and P. Thiran, "A General Framework for Sensor Placement in Source Localization," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 2, pp. 86–102, 2019.

[145] X. Jiang and G. L. Wallstrom, "A Bayesian Network for Outbreak Detection and Prediction," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, p. 1155–1160, AAAI Press, 2006.

[146] X. Jiang, D. Neill, and G. Cooper, "A Bayesian network model for spatial event surveillance," *International Journal of Approximate Reasoning*, vol. 51, pp. 224–239, 2010.

[147] X. Fang, W. Yan, and W. Chen, "Sensor Placement for Underwater Source Localization With Fixed Distances," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 9, pp. 1379–1383, 2016.

[148] L. Cheng, C. Wu, Y. Zhang, H. Wu, M. Li, and C. Maple, "A Survey of Localization in Wireless Sensor Network," *International Journal of Distributed Sensor Networks*, vol. 8, no. 12, p. 962523, 2012.

[149] M. A. Jatoi, N. Kamel, A. S. Malik, I. Faye, and T. Begum, "A survey of methods used for source localization using EEG signals," *Biomedical Signal Processing and Control*, vol. 11, pp. 42–52, 2014.

[150] M. Nourinejad, O. Berman, and R. C. Larson, "Placing sensors in sewer networks: A system to pinpoint new cases of coronavirus," *PLoS One*, vol. 16, Apr. 2021.

[151] E. Calle, D. Martínez, R. Brugués-i-Pujolràs, M. Farreras, J. Saló-Grau, J. Pueyo-Ros, and L. Corominas, "Optimal selection of monitoring sites in cities for SARS-CoV-2 surveillance in sewage networks," *Environment International*, vol. 157, p. 106768, 2021.

[152] E. Domokos, V. Sebestyén, V. Somogyi, A. J. Trájer, R. Gerencsér-Berta, B. Oláhné Horváth, E. G. Tóth, F. Jakab, G. Kemenesi, and J. Abonyi, "Identification of sampling points for the detection of SARS-CoV-2 in the sewage system," *Sustainable Cities and Society*, vol. 76, 2022.

[153] K. Villez, P. A. Vanrolleghem, and L. Corominas, "A general-purpose method for Pareto optimal placement of flow rate and concentration sensors in networked systems – With application to wastewater treatment plants," *Computers & Chemical Engineering*, vol. 139, p. 106880, 2020.

[154] W. Li, J. Han, Y. Li, F. Zhang, X. Zhou, and C. Yang, "Optimal sensor placement method for wastewater treatment plants based on discrete multi-objective state transition algorithm," *Journal of Environmental Management*, vol. 307, p. 114491, 2022.

[155] K. P. Murphy, *Probabilistic Machine Learning: An introduction.* MIT Press, Massachusetts, 2022.

[156] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *ArXiv*, 2010.

[157] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

APPENDIX A: Sensor Placement

A.1 Theory

This section shows the derivation of the SVGP evidence lower bound's delta term, which is used to contrast it with the MI approach's delta term (Section 2.4.3) and details the derivation to shows that our variational formulation of the sensor placement problem and, by extension, the SVGPs are not submodular, which suggests that they are capable of capturing a higher order of information.

A.1.1 Preliminary

A.1.1.1 Properties of Entropy H

1. Joint entropy can be decomposed into the sum of conditional entropy and marginal entropy [3]:

$$\begin{aligned} H(X,Y) =& H(X|Y) + H(Y) \\ =& H(Y|X) + H(X)\,. \end{aligned}$$

2. The reverse KL divergence is the cross entropy minus entropy [155]:

$$\mathrm{KL}(q||p) = H_p(q) - H(q)\,.$$

A.1.1.2 Submodularity

A set function f is submodular if it has the following diminishing returns property for sets X, Y, and T, with u being an element of the set T that is not already in Y [139]:

$$f(X \cup \{u\}) - f(X) \geq f(Y \cup \{u\}) - f(Y)$$
$$\forall X \subseteq Y \subset T \text{ and } u \in T \backslash Y \,.$$

A.1.2 SVGP Evidence Lower Bound's Delta Term Expansion

The lower bound of the SVGP [8] is given by:

$$\mathcal{F} = \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\mathbf{Q}_{nn} + \sigma^2_{\text{noise}} I| + \frac{1}{2}\mathbf{y}^\top (\mathbf{Q}_{nn} + \sigma^2_{\text{noise}} I)^{-1}\mathbf{y} - \frac{1}{2\sigma^2_{\text{noise}}} Tr(\mathbf{K}_{nn} - \mathbf{Q}_{nn}) \,, \tag{A.1}$$

where $\mathbf{K}_{nn}$ is the covariance matrix computed using the SGP's kernel function on the n training samples $\mathbf{X}$, $\mathbf{Q}_{nn} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}$, the subscript m corresponds to the inducing points set, σ_{noise} is the noise variance, and $\mathbf{y}$ is the vector containing the training set labels.

Assume that the inducing points are a subset of the training set indexed by $m \subset \{1, ..., n\}$. Let $(\mathbf{X}_m, \mathbf{f}_m)$ be the set of inducing points locations and their corresponding latent variables. Similarly, let $(\mathbf{X}, \mathbf{f})$ be the training set locations and latent variables. Here n is the index set corresponding to the training dataset, and m is the index set corresponding to the inducing points. Note that we use the same notation as [8], who also used n and m to denote the cardinality of these sets. We know that the SVGP evidence lower bound can be written as follows [3] for inducing points $\mathbf{X}_m$:

$$\begin{aligned} \mathcal{F}(\mathbf{X}_m) &= -\,\text{KL}(q_m(\mathbf{f}) || p(\mathbf{f}|\mathbf{y})) + \log p(\mathbf{y}) \\ &= -\,H_{p(\mathbf{f}|\mathbf{y})}(q_m(\mathbf{f})) + H(q_m(\mathbf{f})) + \log p(\mathbf{y}) \,. \end{aligned} \tag{A.2}$$

Here q_m is the variational distribution of the SGP with the m inducing points. We index the n training set points excluding the m inducing set points as the set

difference $n - m$. We can use the above to formulate the increments (delta term) in the SVGP lower bound upon adding a new inducing point $\mathbf{x}_i$ such that $i \in n - m$ as follows:

$$\begin{aligned}
\Delta\mathcal{F}(\mathbf{X}_m, \{\mathbf{x}_i\}) =& \mathcal{F}(\mathbf{X}_m \cup \{\mathbf{x}_i\}) - \mathcal{F}(\mathbf{X}_m) \\
=& - \mathrm{KL}(q_{m+1}(\mathbf{f})||p(\mathbf{f}|\mathbf{y})) + \mathrm{KL}(q_m(\mathbf{f})||p(\mathbf{f}|\mathbf{y})) \\
=& - H_{p(\mathbf{f}|\mathbf{y})}(q_{m+1}(\mathbf{f})) + H(q_{m+1}(\mathbf{f})) + H_{p(\mathbf{f}|\mathbf{y})}(q_m(\mathbf{f})) - H(q_m(\mathbf{f})) \\
=& \underbrace{(H(q_{m+1}(\mathbf{f})) - H(q_m(\mathbf{f})))}_{\Delta h_1} - \underbrace{(H_{p(\mathbf{f}|\mathbf{y})}(q_{m+1}(\mathbf{f})) - H_{p(\mathbf{f}|\mathbf{y})}(q_m(\mathbf{f})))}_{\Delta h_2} .
\end{aligned} \tag{A.3}$$

The last equation above is similar to the KL divergence, except that each entropy term Δh_j here is the difference of two entropies. We can use the following expansion of the variational distribution q_m to simplify the above:

$$\begin{aligned}
q_m(\mathbf{f}) &= p(\mathbf{f}_{n-(m+1)}, f_i|\mathbf{f}_m)\phi(\mathbf{f}_m) \\
&= p(\mathbf{f}_{n-(m+1)}|f_i, \mathbf{f}_m)p(f_i|\mathbf{f}_m)\phi(\mathbf{f}_m) .
\end{aligned} \tag{A.4}$$

Here we factorized the variational distribution over $\mathbf{f}$ as the product of the variational distribution ϕ over the latents $\mathbf{f}_m$ parametrized with the m inducing points $\mathbf{X}_m$ and the conditional distribution p over the remaining data points $n - m$ computed using conditioning; f_i corresponds to the additional data sample added to the m inducing points. Similar to the above we can expand $q_{m+1}(\mathbf{f})$ as follows:

$$\begin{aligned}
q_{m+1}(\mathbf{f}) &= p(\mathbf{f}_{n-(m+1)}|\mathbf{f}_{m+1})\phi(\mathbf{f}_{m+1}) \\
&= p(\mathbf{f}_{n-(m+1)}|\mathbf{f}_m, f_i)\phi(f_i|\mathbf{f}_m)\phi(\mathbf{f}_m) .
\end{aligned} \tag{A.5}$$

Instead of using the conditional $p(f_i|\mathbf{f}_m)$ as we did for $q_m(\mathbf{f})$, here the distribution over f_i is from the variational distribution $\phi(f_i|\mathbf{f}_m)$. Since all the inducing points are explicitly given in the variational distribution, the joint variational distribution over

the inducing points can be decomposed as the product of marginals. We can now plug the decomposed variational distribution back into Equation A.3 to get the following using the chain rule of entropy:

$$\begin{aligned}
\Delta h_1 =& H(q_{m+1}(\mathbf{f})) - H(q_m(\mathbf{f})) \\
=& H(p(\mathbf{f}_{n-(m+1)}|f_i, \mathbf{f}_m)\phi(f_i|\mathbf{f}_m)\phi(\mathbf{f}_m)) - H(p(\mathbf{f}_{n-(m+1)}|f_i, \mathbf{f}_m)p(f_i|\mathbf{f}_m)\phi(\mathbf{f}_m)) \\
=& \cancel{H(p(\mathbf{f}_{n-(m+1)}|f_i, \mathbf{f}_m))} + H(\phi(f_i|\mathbf{f}_m)) + \\
& \cancel{H(\phi(\mathbf{f}_m))} - \cancel{H(p(\mathbf{f}_{n-(m+1)}|f_i, \mathbf{f}_m))} - H(p(f_i|\mathbf{f}_m)) - \cancel{H(\phi(\mathbf{f}_m))} \\
=& H(\phi(f_i|\mathbf{f}_m)) - H(p(f_i|\mathbf{f}_m)) \,.
\end{aligned} \tag{A.6}$$

Similar to the above, we can get $\Delta h_2 = H_{p(f_i|\mathbf{y})}(\phi(f_i|\mathbf{f}_m)) - H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_m))$. This gives us the following:

$$\begin{aligned}
\Delta\mathcal{F}(\mathbf{X}_m, \{\mathbf{x}_i\}) &= H(\phi(f_i|\mathbf{f}_m)) - H(p(f_i|\mathbf{f}_m)) - H_{p(f_i|\mathbf{y})}(\phi(f_i|\mathbf{f}_m)) + H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_m)) \\
&= (H(\phi(f_i|\mathbf{f}_m)) - H_{p(f_i|\mathbf{y})}(\phi(f_i|\mathbf{f}_m))) - (H(p(f_i|\mathbf{f}_m)) - H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_m))) \\
&= \mathrm{KL}(\phi(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y})) - \mathrm{KL}(p(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y}))\,.
\end{aligned} \tag{A.7}$$

Consider $\mathbf{X}_m \subseteq \mathbf{X}_l \subset \mathbf{X}$ and $\mathbf{x}_i \in \mathbf{X} \backslash \mathbf{X}_l$:

$$\begin{aligned}
\Delta\mathcal{F}(\mathbf{X}_m, \{\mathbf{x}_i\}) - \Delta\mathcal{F}(\mathbf{X}_l, \{\mathbf{x}_i\}) \geq& 0 \\
\mathrm{KL}(\phi(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y})) - \mathrm{KL}(p(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y})) -& \\
\mathrm{KL}(\phi(f_i|\mathbf{f}_l)||p(f_i|\mathbf{y})) + \mathrm{KL}(p(f_i|\mathbf{f}_l)||p(f_i|\mathbf{y})) \geq& 0
\end{aligned} \tag{A.8}$$

One needs to show that the last equation above is true for the SVGP's lower bound to be submodular, which is not necessarily true in all cases. If we constrain the variational distribution ϕ to have a diagonal covariance matrix, the diagonal covariance matrix assumption allows us to drop the variational distribution's dependence on f_m

in Δh_1 and Δh_2. This gives us the following:

$$\begin{aligned}\Delta h_1 &= H(\phi(f_i)) - H(p(f_i|\mathbf{f}_m)) \\ \Delta h_2 &= H_{p(f_i|\mathbf{y})}(\phi(f_i)) - H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_m))\,.\end{aligned} \tag{A.9}$$

Now if we again consider $\mathbf{X}_m \subseteq \mathbf{X}_l \subset \mathbf{X}$ and $\mathbf{x}_i \in \mathbf{X} \backslash \mathbf{X}_l$:

$$\begin{aligned}\Delta\mathcal{F}(\mathbf{X}_m, \{\mathbf{x}_i\}) - \Delta\mathcal{F}(\mathbf{X}_l, \{\mathbf{x}_i\}) &\geq 0 \\ \mathrm{KL}(\phi(f_i)||p(f_i|\mathbf{y})) - \mathrm{KL}(p(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y}))- & \\ \mathrm{KL}(\phi(f_i)||p(f_i|\mathbf{y})) + \mathrm{KL}(p(f_i|\mathbf{f}_l)||p(f_i|\mathbf{y})) &\geq 0 \\ \cancel{H(\phi(f_i))} - H(p(f_i|\mathbf{f}_m)) - \cancel{H_{p(f_i|\mathbf{y})}(\phi(f_i))} + H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_m))- & \\ \cancel{H(\phi(f_i))} + H(p(f_i|\mathbf{f}_l)) + \cancel{H_{p(f_i|\mathbf{y})}(\phi(f_i))} - H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_l)) &\geq 0 \\ H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_m)) - H(p(f_i|\mathbf{f}_m)) - H_{p(f_i|\mathbf{y})}(p(f_i|\mathbf{f}_l)) + H(p(f_i|\mathbf{f}_l)) &\geq 0 \\ \mathrm{KL}(p(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y})) - \mathrm{KL}(p(f_i|\mathbf{f}_l)||p(f_i|\mathbf{y})) &\geq 0 \\ \mathrm{KL}(p(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y})) &\geq \mathrm{KL}(p(f_i|\mathbf{f}_l)||p(f_i|\mathbf{y})) \\ \mathrm{KL}(p(f_i|\mathbf{f}_m)||p(f_i|\mathbf{y})) &\geq \mathrm{KL}(p(f_i|\mathbf{f}_m \cup (\mathbf{f}_l \backslash \mathbf{f}_m))||p(f_i|\mathbf{y}))\end{aligned} \tag{A.10}$$

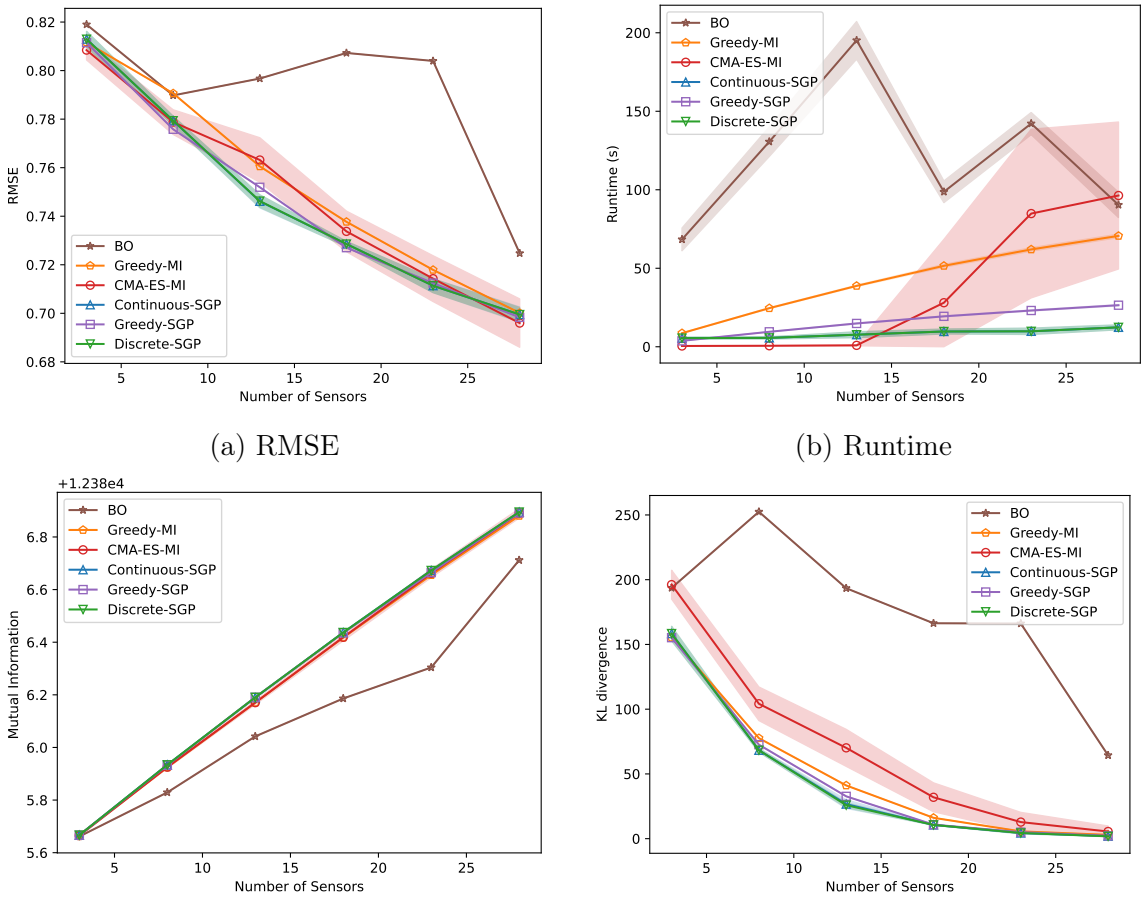
The last equation above is much simpler than the inequality obtained using a full covariance matrix in the variational distribution. Indeed all the distributions used in the KL divergence terms are Gaussian, and therefore have a closed form equation for the KL divergence. However, we still found the inequality far too complex to be able to conclusively prove that the SVGP's lower bound is submodular.

So we performed empirical tests to check for submodularity and found that the SVGP's ELBO with a full covariance matrix and a diagonal covariance matrix were not submodular. Nonetheless, we found that when using a diagonal covariance matrix, the bound was almost submodular. This finding is based on the quality of the solutions found by optimizing the lower bound using the naive greedy and lazy greedy algorithms. When using a diagonal covariance matrix, the solution inducing points'

lower bounds from both the algorithms were very close. This suggests that even if one were to treat the lower bound as submodular and optimize it using the efficient lazy greedy algorithm, we would still get good solutions.

A.2 Bayesian Optimization Results

For completeness, we also generated results on the Intel temperature dataset using Bayesian optimization (BO) [53], shown in Figure A.1. We used the upper confidence bound (UCB) acquisition function [156] in BO and maximized mutual information. However, the method's performance was subpar compared to other baselines.



(a) RMSE

(b) Runtime

(c) Mutual Information

(d) KL Divergence

Figure A.1: Results on the Intel temperature dataset generated using BO and the methods discussed in the chapter—Greedy-MI, CMA-ES-MI, Continuous-SGP, Greedy-SGP, and Discrete-SGP.

APPENDIX B: Informative Path Planning

The following is an example of the expansion transformation operation written as a function in Python with TensorFlow. The function considers sensor with a FoV shaped as a line with a fixed length.

Algorithm 10: Expansion transformation function (written in Python with TensorFlow [157]) used to map the 2D position (x, y) and orientation (θ) to a set of points along a line segment with the origin at the 2D point in the direction of the orientation θ. Here, $\mathbf{X}_m$ are the inducing points with the position and orientation parameterization, l is the length of the line along which the mapped points are sampled, and p is the number of points that are sampled along the line.

Input: $\mathbf{X}_m$, l, p
$x, y, \theta = \text{tf.split}(\mathbf{X}_m, \text{num\_or\_size\_splits} = 3, \text{axis} = 1)$
$x = \text{tf.squeeze}(x)$
$y = \text{tf.squeeze}(y)$
$\theta = \text{tf.squeeze}(\theta)$
$\mathbf{X}_m = \text{tf.linspace}([x, y], [x + l \times \text{tf.cos}(\theta), y + l \times \text{tf.sin}(\theta)], p, \text{axis} = 1))$
$\mathbf{X}_m = \text{tf.transpose}(\mathbf{X}_m, [2, 1, 0])$
$\mathbf{X}_m = \text{tf.reshape}(\mathbf{X}_m, [-1, 2])$
return $\mathbf{X}_m$

The aggregation transformation matrix $T_{\text{agg}} \in \mathbb{R}^{mp \times m}$ is populated as follows for $m = 3$ and $p = 2$ for mean aggregation:

$$T_{\text{agg}}^{\top} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}.$$

However, one can even use the 1-dimensional average pooling operation to efficiently apply the aggregation transformation without having to store large aggregation matrices.

APPENDIX C: Graph Informative Path Planning

C.1 Derivation of EER and Posterior for Gaussian prior

We use the second order Rényi entropy function in our Bayesian model.

$$H(R) = -\log_2 \int_{r \epsilon R} p^2(r) dr \tag{C.1}$$

The conditional probability of a gas concentration given a leak rate is defined as follows. We marginalize this term to get the probability distribution of gas concentration m.

$$p(s|m) = \frac{p(m|s)p(s)}{\int_{s_1 \epsilon S} p(m|s_1)p(s_1)ds_1} \tag{C.2}$$

$$p(m) = \int_{s \epsilon S} p(m,s)ds = \int_{s \epsilon S} p(m|s)p(s)ds \tag{C.3}$$

Albertson et al. [95] modeled the distribution of a leak rate across data collection tours with the following wherein j is the tour number. Furthermore, we model the initial prior as a Gaussian as shown below.

$$p(s) = \begin{cases} p(s), & \text{for } j = 1 \\ p(s|m)_{j-1}, & \text{for } j > 1 \end{cases} \tag{C.4}$$
$$p(s) \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

We, consider the gas dispersion model of Foster et al. [112] in our derivation. This allows us to model the conditional of a gas concentration rate given a leak rate in terms of the dispersion model. Moreover, we assume the dispersion model to be linear

in the leak rate.

$$
\begin{aligned}
C(s,x,y,z) &= s \underbrace{\frac{1}{U}\left(\frac{\bar{A}}{\overline{z}(x)}\exp\left[-\left(\frac{Bz}{\overline{z}(x)}\right)^2\right]\right)\left(\frac{1}{\sqrt{2\pi}\sigma_y}\exp\left[-\frac{1}{2}\left(\frac{y}{\sigma_y}\right)^2\right]\right)}_{\text{constant, independent of function input } s} \\
&= s\mathcal{A}(x,y,z)
\end{aligned}
\tag{C.5}
$$

$$
\begin{aligned}
p(m|s) &= \frac{1}{\sigma_e\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{m-C(s,x,y,z)}{\sigma_e}\right)^2\right] \\
&= \frac{1}{\sigma_e\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{m-s\mathcal{A}(x,y,z)}{\sigma_e}\right)^2\right] \\
p(m|s) &\sim \mathcal{N}(s\mathcal{A},\sigma_e^2)
\end{aligned}
\tag{C.6}
$$

$$
\begin{aligned}
p(m|s)p(s) &= \mathcal{N}(s\mathcal{A}, \sigma_e^2)\mathcal{N}(\mu_s, \sigma_s^2) \\
&= \frac{1}{\sigma_e\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{m - s\mathcal{A}}{\sigma_e}\right)^2\right] \frac{1}{\sigma_s\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{s - \mu_s}{\sigma_s}\right)^2\right] \\
&= \frac{1}{2\pi\sigma_e\sigma_s} \exp\left[-\frac{1}{2}\left(\left(\frac{m - s\mathcal{A}}{\sigma_e}\right)^2 + \left(\frac{s - \mu_s}{\sigma_s}\right)^2\right)\right] \\
&= \frac{1}{2\pi\sigma_e\sigma_s} \exp\left[-\frac{1}{2}\left(\frac{m^2 - 2ms\mathcal{A} + s^2\mathcal{A}^2}{\sigma_e^2} + \frac{s^2 - 2s\mu_s + \mu_s^2}{\sigma_s^2}\right)\right] \\
&= \frac{1}{2\pi\sigma_e\sigma_s} \exp\left[-\frac{s^2 - 2s\frac{m\mathcal{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2} + \frac{\mu_s^2\sigma_e^2 + m^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}}\right] \\
&= \frac{1}{2\pi\sigma_e\sigma_s} \exp\left[-\frac{\left(s - \frac{m\mathcal{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}}\right] \\
&\qquad \exp\left[\underbrace{\frac{\frac{\mu_s^2\sigma_e^2 + m^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2} - \left(\frac{m\mathcal{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}}}_{\Lambda}\right] \\
&= \frac{1}{2\pi\sigma_e\sigma_s} \exp\left[-\frac{\left(s - \frac{m\mathcal{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}}\right] \\
&\qquad \exp\left[\frac{(m - \mu_s\mathcal{A})^2}{2(\mathcal{A}^2\sigma_s^2 + \sigma_e^2)}\right] \quad \text{(From Equation C.12)}
\end{aligned}
\tag{C.7}
$$

The definite integral of an arbitrary Gaussian function

$$
\int_{-\infty}^{\infty} e^{-a(x+b)^2} dx = \sqrt{\frac{\pi}{a}} \tag{C.8}
$$

$$
\begin{aligned}
\int_{-\infty}^{\infty} p^2(s)dx &= \int_{-\infty}^{\infty} (\frac{1}{\sqrt{2\pi\sigma_s^2}} e^{\frac{-(x-\mu_s)^2}{2\sigma_s^2}})^2 dx \\
&= \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_s^2} e^{\frac{-2(x-\mu_s)^2}{2\sigma_s^2}} dx \\
&= \frac{1}{2\pi\sigma_s^2} \int_{-\infty}^{\infty} e^{\frac{-(x-\mu_s)^2}{\sigma_s^2}} dx \\
&= \frac{\sigma_s\sqrt{\pi}}{2\pi\sigma_s^2} \quad \text{(From Equation C.8)} \\
&= \frac{1}{2\sigma_s\sqrt{\pi}}
\end{aligned}
\tag{C.9}
$$

$$
\begin{aligned}
\int\limits_{s\epsilon S} p(m|s)p(s)ds &= \int\limits_{s\epsilon S} \frac{1}{2\pi\sigma_e\sigma_s} \exp\left[-\frac{\left(s - \frac{m\mathcal{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}} \right] \exp\left[\frac{(m - \mu_s\mathcal{A})^2}{2(\mathcal{A}^2\sigma_s^2 + \sigma_e^2)} \right] ds \\
&= \exp\left[\frac{(m - \mu_s\mathcal{A})^2}{2(\mathcal{A}^2\sigma_s^2 + \sigma_e^2)} \right] \frac{1}{2\pi\sigma_e\sigma_s} \\
&\qquad \int\limits_{s\epsilon S} \exp\left[-\frac{\left(s - \frac{m\mathcal{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}} \right] ds \\
&= \exp\left[\frac{(m - \mu_s\mathcal{A})^2}{2(\mathcal{A}^2\sigma_s^2 + \sigma_e^2)} \right] \\
&\qquad \frac{1}{2\pi\sigma_e\sigma_s} \sqrt{\pi \frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2 + \sigma_e^2}} \quad \text{(From Equation C.8)}
\end{aligned}
\tag{C.10}
$$

$$
\begin{aligned}
\int_{s\epsilon S}(p(m|s)p(s))^2 ds &= \int_{s\epsilon S}\left(\frac{1}{2\pi\sigma_e\sigma_s}\exp\left[-\frac{\left(s-\frac{m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}}\right]\right. \\
&\qquad\left.\exp\left[\frac{(m-\mu_s\mathcal{A})^2}{2(\mathcal{A}^2\sigma_s^2+\sigma_e^2)}\right]\right)^2 ds \\
&= \left(\exp\left[\frac{(m-\mu_s\mathcal{A})^2}{2(\mathcal{A}^2\sigma_s^2+\sigma_e^2)}\right]\frac{1}{2\pi\sigma_e\sigma_s}\right)^2 \\
&\qquad\int_{s\epsilon S}\exp\left[-\frac{\left(s-\frac{m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}}\right]^2 ds \\
&= \exp\left[\frac{\cancel{2}(m-\mu_s\mathcal{A})^2}{\cancel{2}(\mathcal{A}^2\sigma_s^2+\sigma_e^2)}\right]\frac{1}{4\pi^2\sigma_e^2\sigma_s^2} \\
&\qquad\int_{s\epsilon S}\exp\left[-\cancel{2}\frac{\left(s-\frac{m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}\right)^2}{\frac{\cancel{2}\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}}\right] ds \\
&= \exp\left[\frac{(m-\mu_s\mathcal{A})^2}{(\mathcal{A}^2\sigma_s^2+\sigma_e^2)}\right]\frac{1}{4\pi^2\sigma_e^2\sigma_s^2} \\
&\qquad\int_{s\epsilon S}\exp\left[-\frac{\left(s-\frac{m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}\right)^2}{\frac{\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}}\right] ds \\
&= \exp\left[\frac{(m-\mu_s\mathcal{A})^2}{(\mathcal{A}^2\sigma_s^2+\sigma_e^2)}\right] \\
&\qquad\frac{1}{4\pi^2\sigma_e^2\sigma_s^2}\sqrt{\pi\frac{\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}} \quad \text{(From Equation C.8)}
\end{aligned}
\tag{C.11}
$$

$$
\begin{aligned}
\Lambda &= \frac{\frac{\mu_s^2\sigma_e^2+m^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2} - \left(\frac{m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}\right)^2}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}} \\
&= \frac{\frac{\mu_s^2\sigma_e^2+m^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2} - \frac{(m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2)^2}{(\mathcal{A}^2\sigma_s^2+\sigma_e^2)^2}}{\frac{2\sigma_e^2\sigma_s^2}{\mathcal{A}^2\sigma_s^2+\sigma_e^2}} \\
&= \frac{\frac{(\mu_s^2\sigma_e^2+m^2\sigma_s^2)(\mathcal{A}^2\sigma_s^2+\sigma_e^2)-(m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2)^2}{(\mathcal{A}^2\sigma_s^2+\sigma_e^2)^{\cancel{2}}}}{\frac{2\sigma_e^2\sigma_s^2}{\cancel{\mathcal{A}^2\sigma_s^2+\sigma_e^2}}} \\
&= \frac{(\mu_s^2\sigma_e^2+m^2\sigma_s^2)(\mathcal{A}^2\sigma_s^2+\sigma_e^2)-(m\mathcal{A}\sigma_s^2+\mu_s\sigma_e^2)^2}{(\mathcal{A}^2\sigma_s^2+\sigma_e^2)(2\sigma_e^2\sigma_s^2)} \\
&= \frac{\mu_s^2\sigma_e^2\mathcal{A}^2\sigma_s^2+m^2\sigma_s^2\sigma_e^2-2m\mathcal{A}\sigma_s^2\mu_s\sigma_e^2}{(\mathcal{A}^2\sigma_s^2+\sigma_e^2)(2\sigma_e^2\sigma_s^2)} \\
&= \frac{\cancel{(\sigma_e^2\sigma_s^2)}(\mu_s^2\mathcal{A}^2+m^2-2m\mathcal{A}\mu_s)}{2(\mathcal{A}^2\sigma_s^2+\sigma_e^2)\cancel{(\sigma_e^2\sigma_s^2)}} \\
&= \frac{(\mu_s^2\mathcal{A}^2+m^2-2m\mathcal{A}\mu_s)}{2(\mathcal{A}^2\sigma_s^2+\sigma_e^2)} \\
&= \frac{(m-\mu_s\mathcal{A})^2}{2(\mathcal{A}^2\sigma_s^2+\sigma_e^2)}
\end{aligned}
\tag{C.12}
$$

The EER was derived as follows:

$$
\begin{aligned}
\varphi[S;M] &= H(S) - \int_{m\epsilon M} H(S|m)p(m)dm \\
&= -\log_2 \int_{s\epsilon S} p^2(s)ds + \int_{m\epsilon M} \log_2 \int_{s\epsilon S} p^2(s|m)ds \; p(m)dm \\
&= -\log_2 \int_{s\epsilon S} p^2(s)ds + \int_{m\epsilon M} \log_2 \int_{s\epsilon S} \left(\frac{p(m|s)p(s)}{\int_{s_1\epsilon S} p(m|s_1)p(s_1)ds_1} \right)^2 ds \; p(m)dm \\
&= -\log_2 \int_{s\epsilon S} p^2(s)ds + \int_{m\epsilon M} \log_2 \frac{\int_{s\epsilon S}(p(m|s)p(s))^2 ds}{(\int_{s_1\epsilon S} p(m|s_1)p(s_1)ds_1)^2} \; p(m)dm \qquad \text{(C.13)} \\
&= -\log_2 \int_{s\epsilon S} p^2(s)ds + \int_{m\epsilon M} \log_2 \frac{\int_{s\epsilon S}(p(m|s)p(s))^2 ds}{(\int_{s_1\epsilon S} p(m|s_1)p(s_1)ds_1)} \; dm \\
&= -\log_2 \left(\frac{1}{2\sigma_s\sqrt{\pi}} \right) + \int_{m\epsilon M} \log_2 \left(\frac{1}{2\sqrt{2}\pi} \exp\left[\frac{(m-\mu_s A_{xyz})^2}{2(A_{xyz}^2\sigma_s^2 + \sigma_e^2)} \right] \right) \; dm \\
&= -\log_2 \left(\frac{1}{2\sigma_s\sqrt{\pi}} \right) + \int_{m\epsilon M} \frac{(m-\mu_s A_{xyz})^2}{2(A_{xyz}^2\sigma_s^2 + \sigma_e^2)} \; dm + c
\end{aligned}
$$

Finally, we derived the posterior of the leak rate as follows.

$$
\begin{aligned}
\psi &= \left(\frac{s-\mu_s}{\sigma_s}\right)^2 + \left(\frac{M-s\mathbf{A}}{\sigma_e}\right)^2 \\
&= \frac{M^\top M - 2M^\top \mathbf{A}s + \mathbf{A}^\top \mathbf{A}s^2}{\sigma_e^2} + \frac{s^2 - 2s\mu_s + \mu_s^2}{\sigma_s^2} \\
&= \frac{M^\top M - 2M^\top \mathbf{A}s + \mathbf{A}^\top \mathbf{A}s^2 + s^2 - 2s\mu_s + \mu_s^2}{\sigma_e^2\sigma_s^2} \\
&= \frac{s^2(\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2) - 2s(M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2) + (\mu_s^2\sigma_e^2 + \sigma_s^2 M^\top M)}{\sigma_e^2\sigma_s^2} \\
&= \frac{s^2 - 2s\frac{M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2} + \frac{\mu_s^2\sigma_e^2 + \sigma_s^2 M^\top M}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}}{\frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}} \\
&= \frac{\left(s - \frac{M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}} + \frac{\frac{\mu_s^2\sigma_e^2 + \sigma_s^2 M^\top M}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2} - \left(\frac{M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}}
\end{aligned}
$$

$$
\begin{aligned}
p(s) &\sim \mathcal{N}(\mu_s, \sigma_s^2) \\
p(M|s) &\sim \mathcal{N}(\mathbf{A}s, \sigma_e^2 I) \\
p(s|M) &= \frac{p(s)p(M|s)}{p(M)} \\
p(s)p(M|s) &= \frac{1}{\sigma_s\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{s-\mu_s}{\sigma_s}\right)^2\right] \frac{1}{\sigma_e\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{M-s\mathbf{A}}{\sigma_e}\right)^2\right] \qquad \text{(C.14)} \\
&= \frac{1}{2\pi\sigma_s\sigma_e} \exp\left[-\frac{1}{2}\left(\underbrace{\left(\frac{s-\mu_s}{\sigma_s}\right)^2 + \left(\frac{M-s\mathbf{A}}{\sigma_e}\right)^2}_{\psi}\right)\right] \\
&= \frac{1}{2\pi\sigma_s\sigma_e} \exp\left[-\frac{1}{2}\left(\frac{\left(s - \frac{M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}} + \right.\right. \\
&\qquad \left.\left. \frac{\frac{\mu_s^2\sigma_e^2 + \sigma_s^2 M^\top M}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2} - \left(\frac{M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}}\right)\right] \\
&= \frac{1}{2\pi\sigma_s\sigma_e} \exp\left[-\frac{1}{2}\frac{\left(s - \frac{M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}}\right] \\
&\qquad \exp\left[-\frac{1}{2}\frac{\frac{\mu_s^2\sigma_e^2 + \sigma_s^2 M^\top M}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2} - \left(\frac{M^\top \mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}\right)^2}{\frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^\top \mathbf{A}\sigma_s^2 + \sigma_e^2}}\right]
\end{aligned}
$$

$$p(s) \sim \mathcal{N}(\mu_s, \sigma_s^2)$$

$$\begin{aligned}
p(m|s) &= \frac{1}{\sigma_e\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{m - C(s,x,y,z)}{\sigma_e}\right)^2\right] \\
C(s,x,y,z) &= s \underbrace{\frac{1}{U}\left(\frac{\bar{A}}{\overline{z}(x)} \exp\left[-\left(\frac{Bz}{\overline{z}(x)}\right)^2\right]\right)\left(\frac{1}{\sqrt{2\pi}\sigma_y} \exp\left[-\frac{1}{2}\left(\frac{y}{\sigma_y}\right)^2\right]\right)}_{\text{independent of leak rate } s} \\
&= s\mathcal{A}(x,y,z) \\
&= \frac{1}{\sigma_e\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{m - s\mathcal{A}(x,y,z)}{\sigma_e}\right)^2\right] \\
p(m|s) &\sim \mathcal{N}(s\mathcal{A}, \sigma_e^2)
\end{aligned} \tag{C.15}$$

$$p(s|M) \propto \mathcal{N}\left(\frac{M^T\mathbf{A}\sigma_s^2 + \mu_s\sigma_e^2}{\mathbf{A}^T\mathbf{A}\sigma_s^2 + \sigma_e^2}, \frac{\sigma_e^2\sigma_s^2}{\mathbf{A}^T\mathbf{A}\sigma_s^2 + \sigma_e^2}\right)$$