Resource Constrained Linear Estimation in Sensor Scheduling and Informative Path Planning

by

Shamak Dutta

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical & Computer Engineering

Waterloo, Ontario, Canada, 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Professor M. Ani Hsieh Dept. of Mechanical Engineering and Applied Mechanics University of Pennsylvania
Supervisor:	Professor Stephen L. Smith Dept. of Electrical and Computer Engineering University of Waterloo
Internal Members:	Professor Christopher Nielsen Dept. of Electrical and Computer Engineering University of Waterloo
	Professor Mark Crowley Dept. of Electrical and Computer Engineering University of Waterloo
Internal-External Member:	Professor Jun Liu Dept. of Applied Mathematics University of Waterloo

Abstract

This thesis studies problems in resource constrained linear estimation with a focus on sensor scheduling and informative path planning. Sensor scheduling concerns itself with the selection of the best subsets of sensors to activate in order to accurately monitor a linear dynamical system over a fixed time horizon. We consider two problems in this setting. First, we study the general version of sensor scheduling subject to resource constraints modeled as linear inequalities. This general form captures a variety of well-studied problems including sensor placement and linear quadratic control (LQG) control and sensing co-design. Second, we study a special case of sensor placement where only k measurements can be taken in a spatial field which finds applications in precision agriculture and environmental monitoring.

In informative path planning, an unknown target phenomena, modeled as a stochastic process, is estimated using a subset of measurements in a spatial field. We study two problems in this setting. First, we consider constraints on robot operation such as tour length or number of measurements with the goal of producing accurate estimates of the target phenomena. Second, we consider the dual version where robots must minimize resources used while ensuring the resulting estimates have low uncertainty or expected squared estimation error.

Our solution approaches exploit the problem structure at hand to give either exact formulations as integer programs, approximation algorithms, or well-designed heuristics that yield high quality solutions in practice. We develop algorithms that combine ideas from combinatorial optimization, stochastic processes, and estimation.

Acknowledgements

I would like to thank Stephen L. Smith for being a fantastic academic advisor. I consider myself extremely fortunate to have had the opportunity to work alongside him the past four years. One of his many great qualities that I particularly admire is his attention to detail. He never overlooked any minor detail and constantly encouraged me to think more clearly about the problem at hand. I am also extremely thankful for his generosity with time and advice. He always entertained conversations on any topic including research, careers, and more often than not, chess.

I am grateful to the doctoral defence committee: Chris Nielsen, Mark Crowley, Jun Liu, and Ani Hsieh, for providing valuable feedback on this thesis.

I have been quite lucky to have taken several courses taught by exceptional professors. I would like to thank Jim Geelen for teaching introduction to optimization in my first semester. Jim taught with extreme clarity and enforced a strict level of technical rigour that I strive to achieve in my work to this day. I would like to thank Stephen Smith for introducing me to stochastic control and estimation. His enthusiasm for teaching and research is infectious, so much so that a large part of this thesis is based on concepts taught in that course. I would also like to thank Levent Tuncel for teaching continuous optimization and Giang Tran for introducing me to functional analysis. While I initially took both these courses for fun, they ended up having a profound impact on my mathematical writing in more ways than one.

Football has been integral to my graduate student life. I would like to thank Mary and Kahtan for organizing our weekly games at Waterloo park. I was often so excited I would not sleep the previous night in anticipation of the fun we would have while kicking a piece of leather around. Thank you Imad, Soufy, Hussein, Isabel, Mehdi, Pranav, Rayan, and Paul for sharing the same excitement. Going further back, I would also like to thank Ryan for inviting me to play in the intramural tournaments with the ECE folks in the first two semesters. We never won any tournaments but scored some great goals together. Thank you Adam, Keegan, Aaron, Orlando, and Madeleine for the memories.

Next, I would like to thank the members of the Autonomous Systems Lab (ASL). It has been extremely rewarding to have Nils as a collaborator on all of my papers. I admire his ability to see the big picture so easily, when working on a research project. I hope we can continue to collaborate in the future. I have also made some of my closest friends by being part of the ASL, also known as the hen pack: Rod, Jack, and Megnath. Some of my fondest memories include our weekly grad-house beer and nachos, jam sessions, go-karting, playing board games together (incomplete without the witty banter), and extremely spicy Bangladeshi food.

I am also thankful to the folks from ‘KW Hangouts’. One of the defining features of this group is that they are always willing to help you out *no matter what*. I must thank Shankha and Harshita for hosting me for nearly a month while apartment hunting. I learnt a great deal about the ‘correct’ bowling action from Shankha during my time at his place. I am also thankful to Harshita for her friendship and the invitations to the impromptu dinners often involving her infamous pani puri. I will always cherish the board game marathons, cryptic crossword attempts, and solving NYT connections with Hemant, Shankha, Shuchita, and Priya. I would be remiss if I did not mention the two best flatmates an individual could ask for: Shuchita and Hopper. Shuchita[1] has been subject to many of my insufferable qualities as a roommate and yet, she has been incredibly supportive and more importantly, a pleasure to be around. Hopper has always provided significant respite from research and never failed to bring a smile to my face. She has taught me the importance of trust and compassion. I know she will never read or understand this but nevertheless, thank you.

I am quite lucky to have been born into a family that has emphasized the importance of a good education. Thank you Mom, Dad, Riteja, and Satya for your love and support. To Mom and Dad, thank you for all the sacrifices you made to ensure Riteja and I had a good upbringing. It is presumptuous to assume a feeble thank you will suffice for everything you have done. I dedicate this thesis to you as a token of my gratitude and hope that it brings you pride.

Finally, I am deeply indebted to Sneha. Among her many great attributes, I am struck by her innate ability to see and hear people deeply, something I am fortunate to have experienced first hand. You have always had my back, thank you. We have grown a tremendous amount together and I am excited to see what the future holds for us.

[1]It is befitting to dedicate a footnote to Cafe Lucero in Kitchener which offers the best third wave coffee in the region. It is safe to say Shuchita and I have spent significant monetary resources enjoying their delicious beverages.

Table of Contents

List of Figures

Chapter 1

Introduction

Linear estimation is a fundamental tool in controls, spatial statistics, and machine learning. For example, it is the bedrock of *Kalman filtering*: the algorithm that revolutionized the field of control theory and was used for navigation in the Apollo project [55]. In spatial statistics, the theoretical underpinnings of *kriging* are rooted in linear estimation and it is used in the prediction of partially observed geological phenomena [33]. Linear estimation also manifests itself in *Gaussian Processes* (GPs), a framework widely used for regression in machine learning [109]. With data proliferation, linear estimation becomes an increasingly important element in the algorithmic toolkit of a scientist.

However, data collection for estimation can be expensive. This is typically the case when hardware components are involved. For example, consider the London Air Quality Network: a collection of air quality monitoring stations spread out in London and South East England [74]. This network comprises of reference grade sensors that are accurate but are expensive to buy and maintain. As a result, it is important to prolong the battery life of these expensive sensors by minimizing the amount of time they are actively collecting data. Data collection can also be hazardous for autonomous systems. In post-disaster environments, there is a need to autonomously map the radiation levels as quickly as possible [26]. The underlying challenge is to efficiently determine which subset of data measurements are critical to the resulting estimates in terms of their expected squared error. In the absence of any constraints, the answer is to use all the data. The presence of resource constraints that show up in practice make this a challenging combinatorial optimization problem. Unfortunately, it is known that the general version of this problem is NP-hard [36] which rules out the existence of polynomial-time algorithms for this problem. In this thesis, we will study this problem and attempt to give algorithms (optimal, approximation, and heuristic) in a variety of special settings.

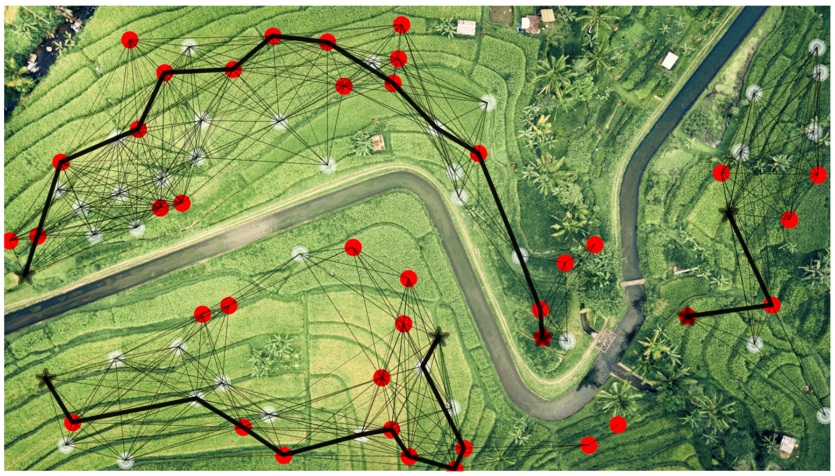


Figure 1.1: The target phenomena to be estimated is the soil nutrient at the red circles in the large spatial field. The red circles are typically given by experts where additional information about the field is desired. The robots plan paths of budgeted length to minimize the expected error at these red circles by collecting data along the black edges. Due to battery constraints, the robots cannot visit all locations and must decide which locations are most useful in minimizing the error.

Broadly speaking, resource constrained linear estimation is concerned with the following problem: given a target phenomena to estimate and resource constraints on data collection, select the subset of data measurements that yield the lowest expected estimation error of the phenomena. For example, in precision agriculture, an important phenomena to estimate is nutrient quality over a large spatial field. A multi-robot team is tasked with collecting soil measurements which are used to map the nutrient quality. However, the robots have finite battery life (resource constraints). Where should the robot team collect data and what paths should they take? We visualize an instance of this problem in Figure 1.1.

We will see that the challenge of resource constrained linear estimation is ubiquitous and shows up in multiple settings including spatial sampling, informative path planning, and sensor scheduling. In the following section, we give a literature review of the approaches taken to solve this challenge and also outline the contributions of this thesis.

1.1 Literature Synopsis

The work in this thesis is related to a large body of work in generalized orienteering, informative path planning, sensor placement/scheduling, and sparse regression. We give a brief literature review here and give an in-depth summary of the most relevant work in the pertaining chapters.

Generalized Orienteering The goal in generalized orienteering is to plan a path of budgeted length that maximizes a set function of the visited vertices. The orienteering problem is a special case when the objective is a modular function of the visited vertices and is known to be NP-hard [54]. When the objective is submodular, a recursive greedy algorithm [24] provides a sub-optimality guarantee but runs in quasi-polynomial time i.e., it scales as $n^{\log n}$ where n is the number of graph vertices. The algorithm has been used in several robotics problems where the objective is submodular [115, 10, 11, 90]. A generalized cost-benefit greedy algorithm was proposed for submodular maximization [142] as well as for the orienteering problem [52]. The popular objective functions used in active regression for GPs are mutual information, which is submodular [78], and the trace of the posterior variance, which is not [36]. Also, the work in [12] proposed a branch and bound algorithm that prunes branches in the search tree using the monotonicity of the posterior variance but is limited to grid graphs with a few vertices. There also exist integer linear programs (ILP) for orienteering [56] and for other related spatial prediction problems (not GP regression) such as correlated orienteering [140].

Informative Path Planning The work in informative path planning in robotics is vast and covers different environment models, error metrics, and robot constraints. We focus our literature review to the GP setting which will be relevant to the work in this thesis. The difficulty in obtaining fast algorithms with optimality or even sub-optimality guarantees for minimizing the posterior variance in GPs has motivated the use of heuristics in the case of planning budgeted paths. One popular heuristic is the greedy algorithm: sequentially select the vertex that yields the maximum marginal increase in utility to the path, normalized by the increase in cost to the path. This has also been applied to provide initial feasible solutions when planning in continuous domains [107, 89, 106, 105]. The choice of the greedy algorithm has been motivated by its application in sensor placement for submodular objectives [78] where it provides sub-optimality guarantees and is computationally efficient. However, it is known that the greedy algorithm can perform arbitrarily poorly when planning paths [115, Section 4]. Finally, the common approach to planning paths for multiple robots is to proceed sequentially i.e., apply the single robot heuristic k times (for k robots) [13]. This yields approximation guarantees for submodular orienteering [115] using the recursive greedy algorithm [24] for each robot. This is also referred to as Sequential Greedy Assignment [3] which was later improved using a distributed algorithm [28] with sub-optimality guarantees using Monte-Carlo Tree Search for each robot. The multi-robot version of orienteering is known as the Team Orienteering Problem [22]. and can be modeled as a mixed integer linear program [56]. Finally, the dual version where the path length is minimized subject to a constraint ensuring the error is below a certain threshold has also been considered [122, 124] which focus on the isotropic kernel setting.

Sensor Placement The sensor placement problem considers the placement of sensors to minimize estimation error subject to a cardinality constraint which is in contrast to path constraints in IPP. The seminal work in sensor placement for GPs showed that the mutual information metric in GPs is submodular [78] thereby motivating the choice of the greedy algorithm which has tight approximation guarantees. Similar greedy approaches have also been used for placement in linear dynamical systems [128, 20] with approximation guarantees and strong empirical performance, although not directly applicable to the GP setting.

Sensor Scheduling Optimizing the subset of output variables to measure in order to best perform state estimation in linear dynamical systems is commonly known as the sensor scheduling problem. It has been studied in various forms in the literature. The difference in formulations generally boils down to the choice of objective function, defined in terms of the Kalman filter error covariance. The work in [69] characterizes the submodularity of commonly used functions such as the trace, maximum eigenvalue, and log determinant.

The trace is not a submodular function and does not benefit from the constant factor approximation guarantees provided by the greedy algorithm. The typical way to address this is to use surrogate objectives such as the log determinant, which is submodular and can be efficiently approximately maximized. However, [20, Remark 1] notes that it is a poor proxy for minimizing the trace. There is a large body of work tackling the log determinant/maximum eigenvalue objective [112, 69, 126, 128, 17, 71, 95] and various other controllability metrics [114, 132, 40]. However, these algorithms do not optimally solve trace minimization over a finite horizon.

The trace minimization of the *steady state* error covariance has been considered [59, 144, 143, 139]. It was established to be NP-hard in [143], which also demonstrated that greedy algorithms perform well in practice but without guarantees. In fact, there is no polynomial time constant factor approximation algorithm for this problem unless P = NP [139].

On a *finite horizon*, greedy algorithms are a popular tool for minimizing the trace of the error covariance due to its computational efficiency and empirical success. Recent work has analyzed their performance through the lens of approximate submodularity [20], weak submodularity [63, 75], and identified strict conditions for submodularity [116]. However, [20, Section VI] notes that the performance bounds tend to be quite loose in practice and greedy algorithms perform near-optimally on small scale instances.

There has been limited work in computing the optimal solution to the sensor scheduling problem. Convex relaxations have provided useful insights [84, 134, 132, 19, 40] for approximate solutions. The work in [131] characterizes certain properties of the optimal solution and uses it to construct a suboptimal solution via tree pruning. The work in [94] formulates a quadratic program with ℓ_0 constraints. The authors solve a relaxed version of the problem and construct an approximate solution through iterative reweighting.

Sparse Regression In sparse regression, one selects a subset of features to minimize the expected error in linear regression. In fact, the exact formulations in this thesis are inspired by the proof of hardness of subset selection for linear regression [36], which performs a reduction from the NP-hard sparse approximation problem [99]. The work in [36] studies a variety of interesting problem instances where sparse regression can either be solved or approximated in polynomial time. Finally, optimally solving the sparse regression problem has been tackled using branch-and-bound [98, 118] and MIPs that can be implemented in modern solvers [5, 8, 7] or using custom branch and bound algorithms [65].

1.2 Thesis Contributions

In this thesis, we study linear estimation in several constrained settings. Since the general version of linear estimation under resource constraints is NP-Hard, we cannot simultaneously have algorithms that a) compute optimal solutions b) in polynomial time c) for any problem instance. The main contributions of this thesis can be summarized as follows:

- We give exact formulations of resource constrained linear estimation in the discrete setting as a mixed integer program. This finds applications in sensor scheduling for linear dynamical systems in Chapter 3 and informative path planning on graphs using GP regression in Chapter 4. We show that by dropping the requirement of polynomial time solvability, optimal solutions can be computed in several interesting settings in seconds.
- We give approximation algorithms in the case when the random variables are indexed by convex sets in Chapter 5. Here, the objective is to minimize the resources used (samples or robot tour lengths) while ensuring the errors resulting from the linear estimators are controlled. The defining features of the algorithms are efficiency, ease of implementation, along with strong theoretical and simulation results.
- Finally, we also give a heuristic in Chapter 6 for the cardinality constrained version of linear estimation where the samples can be taken anywhere in a convex set. While we do not give worst-case performance guarantees, the algorithm performs well in simulation both in terms of solution quality and runtime.

We now review the contributions of each chapter in detail.

Chapter 2: In this chapter, we review the concepts of linear estimation, Gaussian Process regression, variants of the traveling salesman problem, and notions of covering and packing. These will be useful in the design and analysis of the proposed algorithms.

Chapter 3: In this chapter, we consider a general form of the sensor scheduling problem for state estimation of linear dynamical systems, which involves selecting sensors that minimize the trace of the Kalman filter error covariance (weighted by a positive semidefinite matrix) subject to polyhedral constraints. This general form captures several well-studied problems including sensor placement, sensor scheduling with budget constraints, and Linear Quadratic Gaussian (LQG) control and sensing co-design. We present a mixed integer optimization approach that is derived by exploiting the optimality of the Kalman filter. While existing work has focused on approximate methods to specific problem variants, our work provides a unified approach to computing optimal solutions to the general version of sensor scheduling. In simulation, we show this approach finds optimal solutions for systems with 30 to 50 states in seconds.

Chapter 4: In this chapter, we study informative path planning for active regression in Gaussian Processes (GP). Here, a resource constrained robot team collects measurements of an unknown function, assumed to be a sample from a GP, with the goal of minimizing the expected squared estimation error resulting from the GP posterior mean. While greedy heuristics are a popular solution in the case of length constrained paths, there are no known approaches that compute *optimal* solutions in the discrete setting subject to routing constraints. We show that this challenge is surprisingly easy to circumvent. Using the optimality of the posterior mean for a class of functions of the squared loss yields an exact formulation as a mixed integer program (MIP). We demonstrate that this approach can find optimal solutions in a variety of settings in seconds and when terminated early, it finds sub-optimal solutions of higher quality than existing heuristics.

Chapter 5: In this chapter, we consider the sample placement and shortest tour problem for robots tasked with mapping environmental phenomena modeled as Gaussian Processes with isotropic kernels. The goal is to minimize the resources used (data samples or tour length) while ensuring the resulting uncertainty in the estimates (via the posterior variance) is within a given threshold at a set of test locations in the environment. We study both problems in two settings: convex environments with a constant threshold and finite test sets with non-uniform thresholds. This general formulation also captures minimal resource problems for chance constrained classification and regression using Gaussian Processes. We give approximation algorithms in both settings which improve on previous results both in terms of theoretical guarantees and simulations. In addition, we also disprove existing lower bounds provided in the literature for the sample placement problem.

Chapter 6: In this chapter, we consider a subset selection problem in a spatial field where we seek to find a set of k locations whose observations provide the best estimate of the field value at a finite set of prediction locations. The measurements can be taken at any location in the continuous field, and the covariance between the field values at different points is given by the widely used squared exponential covariance function. One approach for observation selection is to perform a grid discretization of the space and obtain an approximate solution using the greedy algorithm. The solution quality improves with a finer grid resolution but at the cost of increased computation. We propose a method to reduce the computational complexity, or conversely to increase solution quality, of the greedy algorithm by considering a search space consisting only of prediction locations and centroids of cliques formed by the prediction locations. We demonstrate the effectiveness of our proposed approach in simulation, both in terms of solution quality and runtime.

Chapter 7: In this chapter, we outline some directions for future research and conclude the thesis.

Chapter 2

Preliminaries

Notation: Let $[n]$ denote the set of positive integers ranging from 1 to n. For a vector $v \in \mathbb{R}^n$, define $\mathrm{supp}(v) := \{i \in [n] : v_i \neq 0\}$. Further, for any $S \subseteq [n]$, let $v_S \in \mathbb{R}^{|S|}$ be the vector with entries v_i for $i \in S$. We denote the set of $n \times n$ positive semidefinite matrices by $\mathbb{S}^n_+$ and positive definite matrices by $\mathbb{S}^n_{++}$. The $n \times n$ identity matrix is denoted by I_n. We use vector/matrix builder notation of the form $A = (a_{ij})_{1\leq i\leq m, 1\leq j\leq n} \in \mathbb{R}^{m\times n}$ where a_{ij} is the entry in row i and column j of A. For a matrix $X \in \mathbb{R}^{m\times n}$, denote the Frobenius norm by $\|X\|_F := (\mathrm{trace}(X'X))^{1/2}$. For two random vectors x, y with dimensions n and m respectively, denote the matrix of covariances between x and y by $\Sigma_{xy} := \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])'] \in \mathbb{R}^{n\times m}$. We denote a metric space by $(\mathcal{X}, \rho)$ where $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a metric. We denote a closed ball of radius r centered at $x \in \mathcal{X}$ by $B(x, r) := \{y \in \mathcal{X} : \rho(x, y) \leq r\}$. For a set A, we denote its boundary by $\mathrm{bd}(A)$.

2.1 Minimum Mean Squared Error (MMSE) Estimation

We begin by introducing estimation in the linear setting. This is the core model used in the thesis and the optimality properties are especially useful in Chapters 3 and 4.

Let x and y be jointly distributed random vectors (of sizes m and n) with zero mean and covariance given by

$$\begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \in \mathbb{S}^{m+n}_+. \tag{2.1}$$

The following characterizes the optimal linear estimator.

Theorem 1 (Chapter 5, Theorem 2.1, [2])**.** The linear least-squares estimator (LLSE) of x given y is given by

$$\hat{x} := K_* y, \tag{2.2}$$

where the optimal coefficient matrix $K_* \in \mathbb{R}^{m\times n}$ is the solution that minimizes the expected squared error

$$K_* := \arg\min_K \mathbb{E}\Big[(x - Ky)' (x - Ky) \Big] = \Sigma_{xy}\Sigma_{yy}^{-1}. \tag{2.3}$$

The resulting error covariance matrix is

$$\mathbb{E}\Big[(x - Ky) (x - Ky)' \Big] = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}. \tag{2.4}$$

The optimal linear estimator satisfies a stronger property given in the following lemma. The proof is relatively straightforward and we include it for completeness.

Lemma 1. For every positive semi-definite matrix $M = Q'Q \in \mathbb{S}_+^n$, the coefficient matrix K_* in (2.3) minimizes the M-weighted expected square error:

$$K_* = \arg\min_K \mathbb{E}\Big[(x - Ky)' M (x - Ky)\Big]. \tag{2.5}$$

Proof. Let $M = Q^T Q$ be arbitrary. Consider the trace of scalar term in the minimization,

$$\begin{aligned}
&\text{trace}\Big(\mathbb{E}\left[(x - Ky)'Q'Q(x - Ky)\right]\Big) = \\
&\quad \text{trace}\Big(Q\mathbb{E}\left[(x - Ky)(x - Ky)'\right]Q'\Big) \\
&\quad = \text{trace}\Big(Q\Big(\Sigma_{xx} - K\Sigma_{yx} - \Sigma_{xy}K' + K\Sigma_{yy}K'\Big)Q'\Big) \\
&\quad = \text{trace}\Bigg(Q\Big(\Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\Big)Q' \\
&\quad + Q\Big(K - \Sigma_{xy}\Sigma_{yy}^{-1}\Big)\Sigma_{yy}\Big(K' - \Sigma_{yy}^{-1}\Sigma_{yx}\Big)Q'\Bigg).
\end{aligned} \tag{2.6}$$

The first term does not depend on K and the second term is non-negative since it is equal to $\|\Sigma_{yy}^{1/2}(K' - \Sigma_{yy}^{-1}\Sigma_{yx})Q'\|_F$. The second term can be made zero by setting $K = \Sigma_{xy}\Sigma_{yy}^{-1}$ which is the coefficient of the optimal linear estimator. □

The optimality of the linear estimator satisfies an even stronger property over the cone of positive semi-definite matrices.

Lemma 2. Let $K \in \mathbb{R}^{m \times n}$ be the coefficients of any linear estimator of the random vector x using the random vector y. Then, the optimal coefficients K_* in (2.3) satisfies the following inequality over the positive semi-definite cone:

$$\mathbb{E}\Big[(x - Ky)\,(x - Ky)' \Big] \succeq \mathbb{E}\Big[(x - K_*y)\,(x - K_*y)' \Big]. \tag{2.7}$$

If the vectors are jointly Gaussian, the linear estimator is the optimal estimator which is a known result in Bayes estimation.

Theorem 2 (Chapter 5, Theorem 2.2, [2])**.** If the vectors x and y are jointly Gaussian, the minimum mean-squared estimator $\mathbb{E}[x|y]$ coincides with the linear least-squares estimator.

2.2 Gaussian Process Regression

Gaussian Processes (GPs) are a popular tool for regression in machine learning. One of the defining features is that the predictors are linear functions of the measurements collected as shown below. We use GPs to model the unknown functions of interest in Chapters 4 and 5.

Definition 1 (Gaussian Process)**.** Let $\mathcal{X}$ be a nonempty set, $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a positive definite function, and $m : \mathcal{X} \to \mathbb{R}$ be any real-valued function. Then, a random function $f : \mathcal{X} \to \mathbb{R}$ is said to be a Gaussian process (GP) with mean function m and covariance kernel k, denoted by $\mathrm{GP}(m, k)$, if the following holds: for any set $X = \{x_1, \ldots, x_n\} \subset \mathcal{X}$, the random vector

$$f_X := (f(x_1), \ldots, f(x_n))' \in \mathbb{R}^n \tag{2.8}$$

follows the multivariate normal distribution $\mathcal{N}(m_X, k_{XX})$ with covariance matrix $k_{XX} = (k(x_i, x_j))_{1 \leq i,j \leq n} \in \mathbb{S}^n_+$ and mean vector $m_X := (m(x_1), \ldots, m(x_n))' \in \mathbb{R}^n$. •

The kernel function k is *isotropic* if it can be formulated as $k(x, y) = h(\|x - y\|)$ for some $h : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ and for any $x, y \in \mathbb{R}^d$. It is common to assume that points that are sufficiently distant are uncorrelated since the covariance decays quickly with the distance between points [78, 18, 30]. In spatial statistics, this is called the *effective or finite range* of the kernel [133]. An isotropic kernel $h : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ has *finite range/compact support* if there exists $r_{\max}$ such that

$$h(\|x-y\|) = 0 \;\; \forall x, y \in \mathbb{R}^d \text{ s.t. } \|x-y\| \geq r_{\max}. \tag{2.9}$$

In GP regression, we estimate an unknown function f, assumed to be a sample from a zero-mean GP[1]

$$f \sim \mathrm{GP}(0, k), \tag{2.10}$$

given noisy measurements of the form

$$y_i := f(x_i) + \eta_i, \tag{2.11}$$

where for $i \geq 1$, η_i are independent and identically distributed $\mathcal{N}(0, \sigma^2)$ Gaussian random variables representing measurement noise and $x_i \in \mathcal{X}$ are the training data inputs. Then, the following well known result shows the conditional predictive distribution of the function is a GP.

Theorem 3 (Theorem 3.1, [73])**.** Assume the setup governed by equations (2.10), (2.11) and let $X := (x_1, \ldots, x_n) \in \mathcal{X}^n$ and $Y := (y_1, \ldots, y_n)^T \in \mathbb{R}^n$ denote the data inputs and measurements respectively. Then, we have

$$f|Y \sim \mathrm{GP}(\bar{m}, \bar{k}), \tag{2.12}$$

where $\bar{m} : \mathcal{X} \to \mathbb{R}$ and $\bar{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are given by

$$\begin{aligned} \bar{m}(x) &:= k_{xX} \left(k_{XX} + \sigma^2 I_n\right)^{-1} Y \\ \bar{k}(x, x') &:= k(x, x') - k_{xX} \left(k_{XX} + \sigma^2 I_n\right)^{-1} k_{Xx'}, \end{aligned} \tag{2.13}$$

where $k_{Xx} = k'_{xX} = (k(x_i, x), \ldots, k(x_n, x))' \in \mathbb{R}^n$. The functions $\bar{m}$ and $\bar{k}$ are known as the *posterior mean function* and *posterior covariance function* respectively. •

For any test inputs $T := \{t_1, \ldots, t_m\} \in \mathcal{X}^m$, the prediction is carried out using the posterior mean and the uncertainty is quantified using the posterior covariance matrix:

$$\begin{aligned} \bar{m}_T^X &:= \mathbb{E}[f_T|Y] = (\bar{m}(t_i))_{i=1}^m \\ &= k_{TX} \left(k_{XX} + \sigma^2 I_n\right)^{-1} Y \in \mathbb{R}^m, \\ \bar{k}_{TT}^X &:= \mathbb{E}\left[(f_T - \bar{m}_T^X)(f_T - \bar{m}_T^X)'|Y\right] = \left(\bar{k}(t_i, t_j)\right)_{i,j=1}^m \\ &= k_{TT} - k_{TX} \left(k_{XX} + \sigma^2 I_n\right)^{-1} k_{XT} \in \mathbb{S}_+^m, \end{aligned} \tag{2.14}$$

where we make the dependence on the training inputs X and test inputs T explicit by using the notation $\bar{m}_T^X$ and $\bar{k}_{TT}^X$.

[1] If the mean function is non-zero, the resulting posterior mean would be *affine* instead of linear.

Remark (Evaluation of Posterior Covariance)**.** It is crucial to note the posterior covariance function is *independent* of the measurements Y; it depends only on the *inputs* X and x through the matrices k_{XX} and k_{Xx}. As a result, the posterior covariance at a test input $x \in \mathcal{X}$ can be computed *a priori* without requiring its associated measurement. •

2.3 Covering & Packing

We introduce covering and packing in continuous and discrete settings. These concepts will be useful in the analysis of the algorithms presented in Chapter 5. Consider a metric space $(\mathcal{X}, \rho)$ and let $A \subset \mathcal{X}$.

Definition 2 (r-packing)**.** The set $\{x_1, \ldots, x_n\}$ is an r-packing of A if the sets $\{B(x_i, r/2) : i \in [n]\}$ are pairwise disjoint i.e., for any $i \neq j$, $\rho(x_i, x_j) > r$.

Definition 3 (r-covering)**.** The set $\{x_1, \ldots, x_n\}$ is an r-covering of A if $A \subset \bigcup_{i=1}^{n} B(x_i, r)$ i.e., $\forall x \in A, \exists i$ such that $\rho(x_i, x) \leq r$.

Problem 1 (`SetCover`)**.** The input to a set cover problem is a pair $(\mathcal{U}, \mathcal{D})$ where $\mathcal{U}$ is a set of of n elements and $\mathcal{D} = \{D_1, \ldots, D_m\}$ is a collection of m subsets of $\mathcal{U}$ such that $\cup_{i=1}^{m} D_i = \mathcal{U}$. The goal is then to find the smallest sub-collection $J \subseteq [m]$ whose union equals $\mathcal{U}$ i.e., $\cup_{i \in J} D_i = \mathcal{U}$.

We will refer to an instance of set cover with universe $\mathcal{U}$ and collection of subsets $\mathcal{D}$ by `SetCover`$(\mathcal{U}, \mathcal{D})$. The greedy algorithm finds a covering of size at most $\log n$ times the optimal [49]. When the subsets $\mathcal{D}$ are generated by geometric regions such as disks, there exists an algorithm that computes a cover that is at most $O(1)$ times the optimal solution [1].

We will consider covering problems with disks in the plane. We give a definition that will be useful for algorithm analysis.

Problem 2 (`SetCover-D`)**.** Given a set of n disks $\mathcal{B} = \{B(x_i, r_i) : i \in [n]\} \subset \mathcal{X} \subseteq \mathbb{R}^2$ and the Euclidean metric, compute a set $S \subset \mathcal{X}$ of minimum size that has non-empty intersection with each disk.

Given a set of disks $\mathcal{B}$, we will denote an instance to set cover with disks in the plane by `SetCover-D`$(\mathcal{B})$.

2.4 Traveling Salesman Problems

We will consider tour planning in discrete and continuous environments in Chapters 4 and 5. As a result, we require the definitions of the TSP (and its variants) along with the associated approximation algorithms.

Metric TSP The input to the metric traveling salesman problem (TSP) is a complete graph $G = (V, E)$, where the vertex set V lies in a metric space $(\mathcal{X}, \rho)$. A cycle is a sequence of distinct vertices $\langle v_1, \ldots, v_n, v_1 \rangle$ and a tour T is a cycle that visits each vertex in the graph exactly once. The cost of a tour, denoted by $\text{length}(T)$, is the sum of the costs of its associated edge set i.e., $\text{length}(T) := \rho(v_n, v_1) + \sum_{i=1}^{n-1} \rho(v_i, v_{i+1})$. The objective of the TSP is then to find a tour of minimum cost. With slight abuse of notation, we let T refer to the tour as well the set of vertices visited on the tour.

TSP with Neighbourhoods The input to the TSP with neighbourhoods (TSPN) is a set of regions in a metric space $(\mathcal{X}, \rho)$. The objective is to find the shortest tour that visits each region at least once. We give a problem definition for disk neighbourhoods which will be useful in our analysis.

Problem 3 (`TSPN-D`)**.** Given a set of n disks $\mathcal{B} = \{B(x_i, r_i) : i \in [n]\} \subset \mathbb{R}^2$ and the Euclidean metric, compute a minimum length tour that visits each disk at least once.

We will denote an instance of the TSP with disk neighbourhoods for a set of disks $\mathcal{B}$ by `TSPN-D`($\mathcal{B}$). There exists an algorithm that computes a tour for `TSPN-D` with length at most $O(1)$ times the optimal solution in time that is polynomial in the number of disks [42].

Chapter 3

Sensor Scheduling for Optimal Kalman Filtering

3.1 Introduction

In many dynamical systems, it is practically infeasible to measure *all* output variables due to communication or energy constraints. Thus, practical solutions often have to rely on only accessing a subset of the data. Optimizing the subset of output variables to measure in order to best perform state estimation is commonly known as the *sensor scheduling problem*. This finds applications in multi-robot environmental monitoring [82, 79, 104], minimal controllability problems [102, 127, 120], control and sensing co-design [125, 141, 113], among others.

Sensor scheduling considers a stochastic linear system evolving in discrete time, which can be partially observed using sensors. Due to resource constraints (e.g., energy constraints), at each time step only a subset of sensors can be turned on. By limiting the number of active sensors, the sensor battery life is prolonged, allowing for long-term usage. The objective is to determine a sensor schedule, i.e., a subset of sensors to be activated at each step, that minimizes the trace of the Kalman filter weighted error covariance while satisfying the resource constraints. The weight is provided by a positive semidefinite matrix which can encode several objectives including final state error and average error. When the chosen sensor set is not allowed to change over time, the problem is known as the *sensor selection* problem. The sensor scheduling problem and its variants are challenging; existing approaches have provided approximate solutions via greedy algorithms [112, 69, 128, 126, 143, 116, 139, 20, 75, 125], convex relaxations

[71, 134, 95, 94, 132, 19, 97, 40, 84], and randomized algorithms [114, 63, 17]. However, in this chapter we strive to compute *optimal* solutions leveraging advances in mixed integer optimization.

While approximation algorithms have yielded useful insights into specific variants of the problem, an approach that computes optimal solutions to the general version has been elusive. The main difficulty lies in the analysis of the Kalman filter recursive equations in terms of the selected sensor subsets. However, the combinatorial nature of the problem suggests encoding the problem as an integer program.

Despite integer programming being NP-complete [39] and generally being considered intractable beyond specific cases, the mixed integer optimization community has made tremendous theoretical and practical advances in last few decades with *machine-independent* speedup factors of up to 580,000 [5, Section 2.1]. One of the benefits is that modern solvers are anytime i.e., if terminated early, approximate solutions with suboptimality certificates are provided to the user. It is relatively straightforward to set up most combinatorial problems as integer programs by modelling the objects of interest as binary decision variables. However, the structure of the resulting objective may not be amenable to optimization. It is well known that the formulation plays an important role in solving integer programs [58, Section 1.2]. This is because the relaxation is used extensively to provide lower bounds in the branch and bound procedure in mixed integer optimization. Thus, if the relaxation has good structure (linearity, convexity) and is tight, one can find optimal solutions relatively quickly by pruning many parts of the search tree. The question we wish to answer is whether one can leverage the structure of the sensor scheduling problem to formulate mixed integer programs.

3.1.1 Contributions

Our main contribution is the formulation of the general sensor scheduling problem as a mixed integer program with a convex quadratic objective and linear constraints (Theorem 4). This enables us to leverage modern solvers such as Gurobi [60] or CPLEX [32]. We accomplish this using the optimality property of the Kalman filter for the trace of a weighted error covariance. This enables us to optimize over the class of linear filters, resulting in a convex minimization problem. This circumvents the need to work with the recursive form of the filter and the associated non-linear algebraic Riccati equations, which can be difficult to model. In simulations, our approach computes optimal solutions to sensor selection and scheduling problems on systems with 35 to 50 states in seconds.

3.2 Problem Formulation

Our setup closely follows [94] which captures a variety of interesting sensor scheduling problems.

Dynamical System: Consider the following linear system:

$$\begin{aligned} x_{k+1} &= Ax_k + w_k, \\ y_k &= Cx_k + v_k, \end{aligned} \tag{3.1}$$

where the process noise w_k, measurement noise v_k, and initial state x_1 are zero mean uncorrelated random variables. For all $k \geq 1$, we assume the covariance of w_k and v_k are given by $W \in \mathbb{S}^n_{++}$ and $V \in \mathbb{S}^m_{++}$ respectively and the *a priori* covariance of the initial state x_1 is given by $\Sigma_{1|0} \in \mathbb{S}^n_{++}$. Further, we assume the state $x_k \in \mathbb{R}^n$ and the vector resulting from all m sensor measurements is $y_k \in \mathbb{R}^m$.

Sensor Schedule: For a given time horizon T, the total number of measurements is mT since each step yields m sensor readings. For $i \in [mT]$, consider binary indicator variables $\gamma_i \in \{0, 1\}$ such that if sensor $j \in [m]$ is on at time step $k \in [T]$, then $\gamma_{m(k-1)+j} = 1$. The binary vector $\gamma = [\gamma_1, \ldots, \gamma_{mT}] \in \{0, 1\}^{mT}$ is called a *sensor schedule*.

Kalman filter: The Kalman filter (KF) is usually described recursively in terms of the estimates from the previous time step. However, viewing it directly as a linear estimator over the set of sensor measurements will be more useful for our purposes. Let $\gamma \in \{0, 1\}^{mT}$ be a sensor schedule over a horizon T with $S_\gamma := \operatorname{supp}(\gamma)$. For each $k \in [T]$, define

$$\begin{aligned} Y_k &:= \left[y_1', \ldots, y_k'\right]' \in \mathbb{R}^{mk} \ \left(\text{measurements until time } k\right) \\ \hat{x}^\gamma_{k|k} &:= K^\gamma_{*,k} Y_{k,\gamma} \in \mathbb{R}^n \ \left(\text{KF estimate using selected measurements}\right) \\ e^\gamma_k &:= x_k - \hat{x}^\gamma_{k|k} \in \mathbb{R}^n \ \left(\text{KF estimation error}\right) \\ \Sigma^\gamma_{k|k} &:= \Sigma_{e^\gamma_k e^\gamma_k} \in \mathbb{S}^n_+ \ \left(\text{KF posterior error covariance}\right) \end{aligned} \tag{3.2}$$

Note that $Y_{k,\gamma} \in \mathbb{R}^{|S_\gamma|}$ only uses the measurement variables selected by the sensor schedule γ. Further, $K^\gamma_{*,k} \in \mathbb{R}^{n \times |S_\gamma|}$ contains the optimal Kalman filter coefficients for estimating the state x_k using measurement variables $Y_{k,\gamma}$. This is different from the gain matrix used in the recursive form of the filter. Each row $i \in [n]$ of $K^\gamma_{*,k}$ contains the coefficients of the optimal linear combination of the measurement variables used in estimating the i^{th} entry of the state x_k.

In summary, given the schedule γ, the Kalman filter computes the optimal linear estimate $\hat{x}^\gamma_{k|k}$ of the state x_k and provides the *a posteriori* error covariance estimate $\Sigma^\gamma_{k|k}$.

Our goal is to compute the schedule γ that minimizes a function related to $\Sigma^{\gamma}_{k|k}$, subject to resource constraints on γ.

Problem 4 (Sensor Scheduling)**.** Given a time horizon $T > 0$, cost matrices $Q_k \in \mathbb{R}^{n' \times n}$ with $M_k := Q_k' Q_k \in \mathbb{S}^n_+$, constraint matrix $H \in \mathbb{R}^{h \times mT}$, and budget $b \in \mathbb{R}^h$, compute the sensor schedule γ that solves

$$\begin{aligned} \underset{\gamma \in \{0,1\}^{mT}}{\text{minimize}} \quad & \sum_{k=1}^{T} \text{trace}\left(Q_k \Sigma^{\gamma}_{k|k} Q_k'\right) \\ \text{subject to} \quad & H\gamma \leq b. \end{aligned} \tag{3.3}$$

The objective and constraints capture several interesting problems. For example, consider the following objectives.

1. *Total Error*: $Q_k = \theta_k \mathbb{I}_n, k \in [T]$, where $\theta_k \in \mathbb{R}_{\geq 0}$.
2. *Final State Error*: $Q_1 = \ldots = Q_{T-1} = 0, Q_T = \mathbb{I}_n$.
3. *LQG Sensing Design*: It is known that the optimal strategy for LQG control is to provide state estimates via Kalman filtering and then perform LQR control using the state estimate in place of the true state. A similar principle holds in LQG control and sensing co-design [125, Theorem 1] where one designs a sensor selection policy as well as the optimal controller. The result states the budgeted sensor selection problem can be decoupled from the control design where the sensor schedule is computed to optimize trace$(\Theta \Sigma^{\gamma}_{k|k})$, and Θ is computed from the LQG system parameters.

Further, the polyhedral constraints in $H\gamma \leq b$ are general and capture different types of resource constraints.

1. *Sensor Selection*: One commits to a sensor subset of size p for all steps: $\sum_{i=1}^{m} \gamma_i = p$ and $\gamma_i = \gamma_{i+jm}$, for $i \in [m], j \in [T-1]$ effectively using only m variables.
2. *Sensor Scheduling*: One selects a sensor subset for each time step $k \in [T]$ of size p: $\sum_{i=1}^{m} \gamma_{(k-1)m+i} = p$.
3. *Energy Constraints*: If sensor i incurs an energy cost α_i each time it is switched on and has an energy budget β_i, then the constraint can be represented as $\sum_{k=1}^{T} \gamma_{(k-1)m+i} \leq \beta_i / \alpha_i$.

Various problems in the literature are modeled as combinations of these objectives and constraints. For example, objective 1) and constraint 1) model the *sensor selection* problem [134, 19, 69, 128, 116, 20, 75, 63]. Combining the accumulated error in objective 2) and constraint 2) is sensor scheduling with budget constraints [131, 84].

3.3 Mixed Integer Program for Sensor Scheduling

This section describes our MIQP formulation. As discussed in the introduction, one requires good formulations in mixed integer programming to compute optimal solutions relatively quickly [58]. For example, if the objective is convex (when the binary variables are relaxed), one can quickly solve the ensuing convex optimization problem to get a lower bound on the optimal solution to the mixed integer program. This helps in eliminating parts of the search tree by comparing lower bounds (by solving the relaxed problem) and upper bounds given by feasible solutions. In the context of sensor scheduling, it is not immediately obvious how one would model the objective in (3.3) as a convex function.

The convex MIQP formulation is presented in Section 3.3.1 with Theorem 4 giving the main result. Note that the results in Section 3.3.1 depend on the covariance matrices between the states and observations in system (3.1), which are fully specified in Section 3.3.2.

3.3.1 Binary Convex Reformulation

We wish to rewrite the objective in (3.3) purely in terms of continuous variables. By exploiting the optimality of the Kalman filter, we reformulate the summands in (3.3) as convex minimization problems where the continuous variables are the coefficients of linear filters. The key idea is to optimize over the class of linear filters using *all* sensor measurement variables with the constraint that a measurement variable i can only be used if its corresponding entry in the schedule satisfies $\gamma_i = 1$. This is established in Lemma 3.

Lemma 3. Given a schedule $\gamma \in \{0,1\}^{mT}$ and a time step $k \in [T]$, each summand in (3.3) can be represented as the following optimization problem

$$\begin{aligned} \operatorname{trace}\left(Q_k \Sigma^{\gamma}_{k|k} Q'_k\right) = \min \Big\{ c_k(K) : \ & [K]_{ij}(1-\gamma_j) = 0, \\ & i \in [n], j \in [mk], \\ & K \in \mathbb{R}^{n \times mk} \Big\}, \end{aligned} \tag{3.4}$$

where $c_k : \mathbb{R}^{n\times mk} \to \mathbb{R}_{\geq 0}$ is the expected squared error of a linear filter specified by coefficients K and is defined as

$$\begin{aligned} c_k(K) &:= \mathbb{E}\Big[(x_k - KY_k)' \, M_k \, (x_k - KY_k) \Big] \\ &= \operatorname{trace}\Big(M_k \Big(K\Sigma_{Y_kY_k}K' - 2\Sigma_{x_kY_k}K' + \Sigma_{x_kx_k} \Big) \Big). \end{aligned} \tag{3.5}$$

Proof. Let $S_\gamma := \operatorname{supp}(\gamma) \cap [mk]$. Consider the RHS in (3.4). The constraint gives the following equivalence: $\gamma_j = 0 \implies K_j = 0$ i.e., the j^{th} column of K is the zero vector. This implies that when taking a linear combination KY_k, the measurements in Y_k corresponding to the zero column vectors in K can be dropped. Specifically, let $K^\gamma \in \mathbb{R}^{n\times|S_\gamma|}$ be the submatrix of K with the column set $\overline{S_\gamma}$ removed and let $Y_{k,\gamma} \in \mathbb{R}^{|S_\gamma|}$ be the measurement vector constructed by dropping the entries corresponding to $\overline{S_\gamma}$ in Y_k. Then, we can rewrite the minimization problem in (3.4) as

$$\min_{K\in\mathbb{R}^{n\times|S_\gamma|}} \Bigg\{ \mathbb{E}\Big[(x_k - K^\gamma Y_{k,\gamma})' M_k (x_k - K^\gamma Y_{k,\gamma}) \Big] \Bigg\}. \tag{3.6}$$

Now, we can apply Lemma 1 which tells us the solution to the above minimization problem is the optimal linear estimator of x_k given $Y_{k,\gamma}$ i.e., the Kalman filter with coefficient matrix $K^\gamma_{*,k} \in \mathbb{R}^{n\times|S_\gamma|}$. Using the definitions in (3.2), the resulting error is obtained by plugging in $K^\gamma_{*,k}$ and taking the trace of the scalar term in (3.6) yielding

$$\begin{aligned} &\mathbb{E}\Big[(x_k - K^\gamma_{*,k} Y_{k,\gamma})' Q_k' Q_k (x_k - K^\gamma_{*,k} Y_{k,\gamma}) \Big] \\ &= \operatorname{trace}\Big(Q_k \Sigma^\gamma_{k|k} Q_k' \Big), \end{aligned} \tag{3.7}$$

where we have used the property $\operatorname{trace}(AB) = \operatorname{trace}(BA)$ for any conformable matrices A, B. □

Note that we delay the definition of the covariance matrices appearing in c_k until Section 3.3.2 since it is not crucial to understanding the formulated MIQP. The interested reader can refer directly to Equation (3.14) for the exact definition of the covariance matrices. Next, we show that c_k is convex.

Lemma 4 (Convexity). The function $c_k : \mathbb{R}^{n\times mk} \to \mathbb{R}_{\geq 0}$ defined in Lemma 3 is convex.

Proof. To prove convexity, define the following matrix functions $g : \mathbb{S}^n \to \mathbb{R}$ and $h : \mathbb{R}^{n\times mk} \to \mathbb{S}^n$,

$$\begin{aligned} g_k(X) &:= \operatorname{trace}(M_k X) \\ h(K) &= K\Sigma_{Y_kY_k}K' - \Sigma_{x_kY_k}K' - K\Sigma_{Y_kx_k} + \Sigma_{x_kx_k}. \end{aligned} \tag{3.8}$$

Then, we have $c_k = g_k \circ h$. Using a composition theorem for convexity, c_k is convex if g is convex and non-decreasing and h is convex [14, Section 3.6.2]. Convexity of g follows from the linearity of the trace operator and since $M_k \in \mathbb{S}^n_+$, it is non-decreasing [14, Example 3.46]. Since $\Sigma_{Y_kY_k} \in \mathbb{S}^m_+$, h is also convex [14, Example 3.49]. □

Finally, we establish that the sensor scheduling problem (3.3) can be represented as a mixed integer convex quadratic program, which can be tackled by solvers like GUROBI [60].

Theorem 4 (MIQP for Sensor Scheduling)**.** The sensor scheduling problem (3.3) can be formulated as the following mixed-integer convex optimization problem

$$\begin{aligned} \min_{\substack{\gamma\in\{0,1\}^{mT} \\ K_k\in\mathbb{R}^{n\times mk}}} \quad & \sum_{k=1}^{T} c_k(K_k) \\ \text{subject to} \quad & H\gamma \le b, \\ & [K_k]_{ij}\,(1-\gamma_j) = 0, i \in [n],\ j \in [mk],\ k \in [T] \end{aligned} \tag{3.9}$$

where the functions $c_k(\cdot)$ are given by Lemma 3.

Proof. Since each state vector x_k requires a set of coefficients $K_k \in \mathbb{R}^{n\times mk}$, the result follows from the problem definition (3.3) and Lemma 3. Since convexity is preserved under addition, the objective in (3.9) is convex by Lemma 4. □

The optimization problem (3.9) is a MIQP of the sensor scheduling problem where the decision variables are the filter coefficients K_k and binary sensor schedule γ. The last constraint allows a filter coefficient to be used only if the corresponding sensor variable is turned on. This constraint can be implemented as a SOS Type-1 constraint in GUROBI or CPLEX. It should be noted that the number of binary variables is mT and the number of continuous variables is $\sum_{k=1}^{T} nmk = nm\frac{T(T+1)}{2}$. For high-dimensional systems and long horizons, this can quickly become intractable. It would be desirable to have a minimal formulation purely in terms of binary variables γ, which we leave to future work.

Remark (Comparison with [94])**.** At first glance, Theorem 4 seems similar to the quadratic program in [94]. However, the key difference is the relationship between the continuous and binary variables. In Theorem 4, there is one indicator constraint for each entry of the matrix K_k which is easy to implement in modern solvers and also leads to fast solve times as evidenced in Section 3.4. In [94], the constraint is more involved. It has the form $g(X) \leq \gamma_i$, where γ_i is a binary decision variable and g is the ℓ_0 norm of the sum of ℓ_1 norms of the columns of the matrix of continuous variables X. This constraint cannot be directly implemented in modern mixed integer solvers. One has to setup indicator variables to model the relationship between γ_i and the ℓ_0 norm. Unfortunately, this does not yield fast solve times as shown in Section 3.4. It should be noted that [94] did not set out to solve an integer program; it relaxed the formulation to compute approximate solutions using an iterative weighting technique which yielded good solutions in simulations.

3.3.2 Covariance Matrices

The last piece is to specify the covariance matrices used in the definition of the functions c_k in (3.5). First, we require a representation of state x_j in terms of the initial state x_1.

Observation 1 (State Representation)**.** For any $j \in \mathbb{N}$,

$$x_j = A^{j-1}x_1 + \sum_{i=1}^{j-1} A^{j-i-1}w_i, \tag{3.10}$$

Observation 2 (Zero-mean States)**.** For any $j \in \mathbb{N}$, $\mathbb{E}(x_j) = 0$. This follows from the combination of Observation 1 and the fact that the initial state x_0 and process noise w_t are zero-mean random variables.

Using these two observations, we will establish the structure of the covariance between any two states x_i and x_j.

Observation 3 (Covariance between States)**.** For $j \leq i$, the covariance between states x_i

and x_j is given by

$$\begin{aligned}\Sigma_{x_i x_j} &= \mathbb{E}\Bigg[\Big(A^{i-1}x_1 + \sum_{t=1}^{i-1} A^{i-t-1} w_t\Big) \\ &\quad \Big(A^{j-1}x_1 + \sum_{t=1}^{j-1} A^{j-t-1} w_t\Big)'\Bigg] \in \mathbb{R}^{n\times n} \\ &= \mathbb{E}\Big(A^{i-1}x_1 x_1' A^{j-1'} + \sum_{t=1}^{j-1} A^{i-t-1} w_t w_t' A^{j-t-1'}\Big) \\ &= A^{i-1}\Sigma_{1|0}A^{j-1'} + \sum_{t=1}^{j-1} A^{i-t-1} W (A^{j-t-1})'.\end{aligned} \tag{3.11}$$

For $j > i$, $\Sigma_{x_i x_j} = \Sigma'_{x_j x_i}$.

Since the measurement is a linear combination of the state (3.1), we can determine the covariance between any state and measurement variable and also between measurement variables. This is described in the following observations.

Observation 4 (Covariance between State and Measurement)**.** For $j \leq t$, the matrix of covariances between state $x_t \in \mathbb{R}^n$ and all sensor observation $y_j \in \mathbb{R}^m$ is given by

$$\begin{aligned}\Sigma_{x_t y_j} &= \mathbb{E}\Big(x_t (C x_j + v_j)'\Big) \\ &= \Sigma_{x_t x_j} C' \in \mathbb{R}^{n\times m}\end{aligned} \tag{3.12}$$

Observation 5 (Covariance between Measurements)**.** The covariance between measurements y_i and y_j is given by

$$\begin{aligned}\Sigma_{y_i y_j} &= \mathbb{E}\Big((C x_i + v_i)(C x_j + v_j)'\Big) \\ &= C\ \Sigma_{x_i x_j} C' + \mathbb{E}(v_i v_j') \in \mathbb{R}^{m\times m}\end{aligned} \tag{3.13}$$

If $i \neq j$ then $\mathbb{E}(v_i v_j') = 0$ (Kalman filter assumptions) and is equal to $V \in \mathbb{S}^m_{++}$ (noise covariance) otherwise.

We now establish the form of the matrices in Theorem 4 whose entries are given by the observations above.

$$\begin{aligned}\Sigma_{x_k Y_k} &= \begin{bmatrix}\Sigma_{x_k y_1} & \dots & \Sigma_{x_k, y_k}\end{bmatrix} \in \mathbb{R}^{n\times mk}, \\ \Sigma_{Y_k Y_k} &= \begin{bmatrix}\Sigma_{y_1 y_1} & \dots & \Sigma_{y_1 y_k} \\ \vdots & \ddots & \vdots \\ \Sigma_{y_k y_1} & \dots & \Sigma_{y_k y_k}\end{bmatrix} \in \mathbb{S}^{mk}_{++}.\end{aligned} \tag{3.14}$$

By substituting (3.14) into (3.5), the MIQP provided in Theorem 4 is now fully specified. This forms our approach to solving the general sensor scheduling problem.

3.4 Numerical Results

To evaluate our proposed MIQP, we consider two variants involving budget constraints: sensor selection (Section 3.4.1) and sensor scheduling (Section 3.4.2). For both problems, we compare against

- Mo *et al.* [94]: We convert the quadratic program using an ℓ_0 constraint to a MIQP using an indicator constraint (see Section 3.3.1 for a detailed discussion).
- Greedy algorithm: We keep selecting sensors one at a time until the budget constraint is met. In each step, the sensor yielding the highest reduction in error is chosen. For more details, see [69, 20].

We now describe the system parameters. The entries of the process matrix $A \in \mathbb{R}^{n\times n}$ are drawn from a uniform distribution on $[0,1]$ and normalized so that the magnitude of the largest eigenvalue is 0.5 (considering stable systems with long horizons). The entries of the measurement matrix $C \in \mathbb{R}^{m\times n}$ are drawn from a uniform distribution on $[0,1]$. The measurement noise matrix is constructed as $W = LL' \in \mathbb{S}^n_+$ where $L \in \mathbb{R}^{n\times n}$ is drawn from a uniform distribution on $[0,1]$. Finally, the noise matrix is $V = \sigma^2\mathbb{I}_m$ where $\sigma^2 = 0.01$ and the initial state covariance is $\Sigma_{1|0} = \mathbb{I}_n$. We run each algorithm on 20 trials with each trial drawing a new set of system parameters. The experiments are implemented in Gurobi [60] on an AMD Ryzen 7 2700 8-core processor with 16 GB of RAM.

For both selection and scheduling, the goal is to minimize the final state estimation error. Thus, the cost matrices $Q_k \in \mathbb{R}^{n\times n}$ are defined as $Q_k = 0$ for $k \in [T-1]$ and $Q_T = \mathbb{I}_n$. The definitions of the constraint matrix and vector are given in Section 3.2. We now proceed to discuss the results.

3.4.1 Sensor Selection with Budget Constraints

The first question we seek to answer is how the runtime of the MIQP scales as the system size increases. We set a timeout of 120 seconds for each algorithm. The results of this experiment are shown in Figure 3.1. We see that the approach of [94] does not scale favourably and it times out on systems with $n > 30$. On the other hand, the proposed

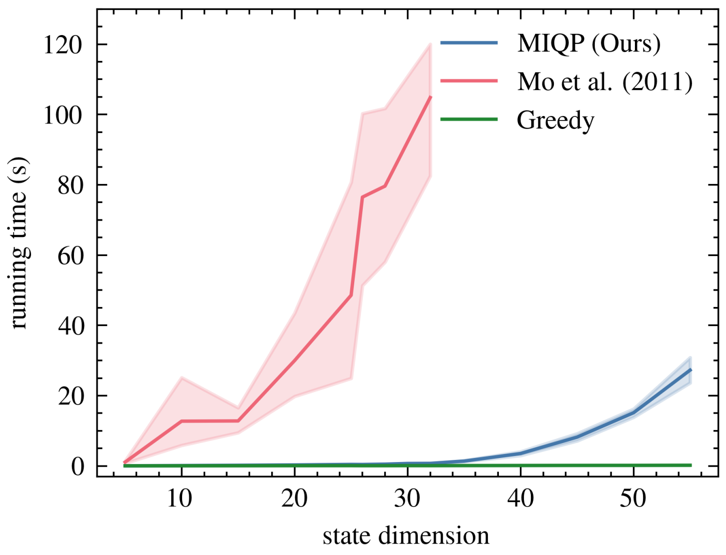


Figure 3.1: **Sensor Selection**: Runtime comparison of our MIQP, [94], and the greedy algorithm. The parameters are $T = 3$ with $m = 10$ sensors and a budget of $p = 5$. The $x-$axis indicates the state dimension while the $y-$axis is the running time measured in seconds.

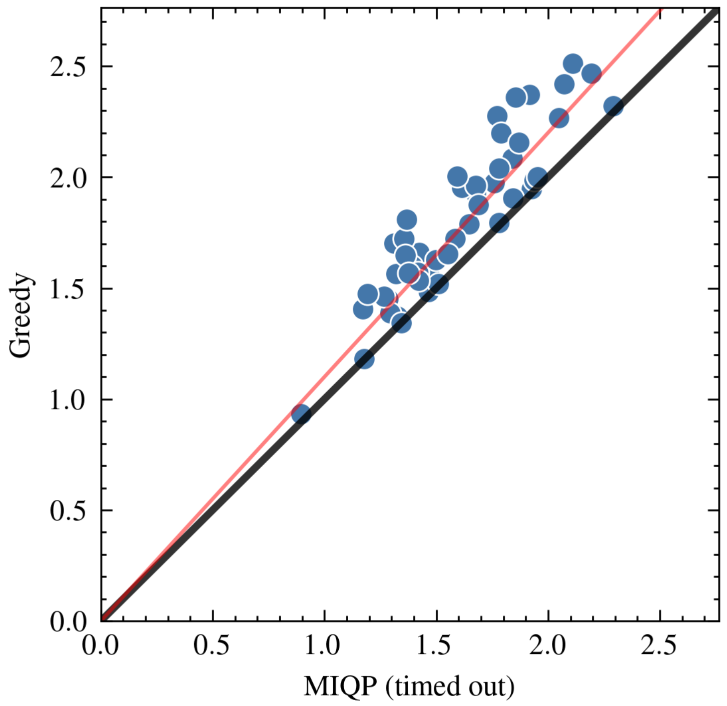


Figure 3.2: **Sensor Selection**: Solution quality comparison of our MIQP (timed out after 10 seconds) and the greedy algorithm. The x and y coordinate denote the solution costs of the MIQP (timed out) and the greedy approach respectively. The points above the black line indicate MIQP obtaining better solutions than greedy. The greedy solution of points above the red line are at least 10% worse than the MIQP solution.

MIQP solves these instances in under a second which is on average, roughly 50 times faster for a system with 25 states. However, the proposed MIQP also shows an increase in runtime as the system size increases albeit at a much slower rate. In fact, until systems of size $n = 35$, the runtime is comparable with greedy. The runtime of greedy is extremely efficient solving all instances in under a second. Further, the approach of [94] exhibits high variance in its runtime which is in contrast to the proposed MIQP and the greedy algorithm which exhibit low runtime variability. In summary, the MIQP has a better runtime when compared to [94] with timings up to 40 times faster while providing *optimal* solutions.

We also investigate the quality of the solution returned by the MIQP when given a time out of 10 seconds. In this situation, it would be apt to compare it to greedy since both return approximate solutions. The setup for this experiment involves fixing the state dimension $n = 10$, horizon $T = 3$, number of sensors $m = 25$, and budget $p = 5$. We select a time out of 10 seconds as it is practical assumption on the runtime for an algorithm. We draw 50 random instances of the system parameters and run both algorithms to get their solutions. We observed that the MIQP timed out on *all* drawn instances. For each instance i, we plot (x_i, y_i) where x_i is the objective value of the solution returned by the MIQP and y_i is the objective value of the greedy solution on instance i. We plot these points in Figure 3.2. We see that all points lie above the black line $y = x$ indicating that even in this challenging setting, the MIQP outperforms greedy on all instances even though it times out. The red line is the function $y = 1.1x$ and thus points that lie above it are instances where the greedy obtains solutions that are at least 10% worse than the MIQP solution. We recorded the mean error difference to be 13%. To summarize, these results suggest the 10 second timed out MIQP returns solutions of quality better or equal to that of the greedy algorithm. The added benefit is that the MIQP gives a certificate of suboptimality even though it times out whereas the greedy algorithm, in general, cannot.

3.4.2 Sensor Scheduling with Budget Constraints

We now study the runtime of each approach for sensor scheduling problems. The results are shown in Figure 3.3. Note that sensor scheduling is a more challenging problem than sensor selection as one has to select a sensor subset for each time step. We see the approach of [94] can only tackle systems of size up to $n = 14$ before timing out (set to 100 seconds). In contrast, our MIQP solves the same instances in under a second giving improvements of up to 40 to 60 times. The greedy remains extremely efficient and provides solutions in under a second for all instances. The summary here is that our MIQP can tackle systems of size up to $n = 35$ while providing optimal solutions before starting to display an increase in runtime, demonstrating its effectiveness.

Our final experiment compares the solution quality of the MIQP, with a 10 second time out, to the greedy algorithm. This follows the same setup used in the second experiment for sensor selection (Section 3.4.1) which generates 50 random instances. In this setting, the MIQP was observed to time out on *all* drawn instances. The results are shown in Figure 3.4. The observations are similar to that in sensor selection. We note the MIQP always returns solutions whose quality is better or equal to that of the greedy algorithm. In addition, the points above the red line are the instances where the greedy solution is at least 10% worse. When compared to the results in sensor selection (Figure 3.2), the number of points above the red line is greater in Figure 3.4. This indicates the performance difference between greedy and the MIQP is larger in sensor scheduling which is also evidenced by the average error difference recorded to be 19%.

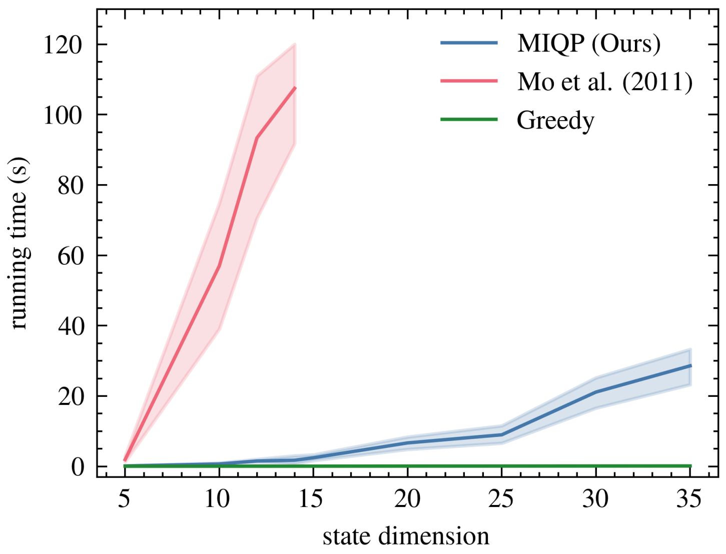


Figure 3.3: **Sensor Scheduling**: Runtime comparison of different algorithms on varying system sizes. The system parameters are $T = 3$, $m = 10$, and budget $p = 5$. The runtime is measured in seconds indicated on the y-axis.

3.5 Summary

We studied the general version of sensor scheduling in linear dynamical systems. We proposed a mixed integer optimization problem that computed optimal solutions. In simulations, we showed the effectiveness of the approach over prior work in exact formulations and greedy algorithms.

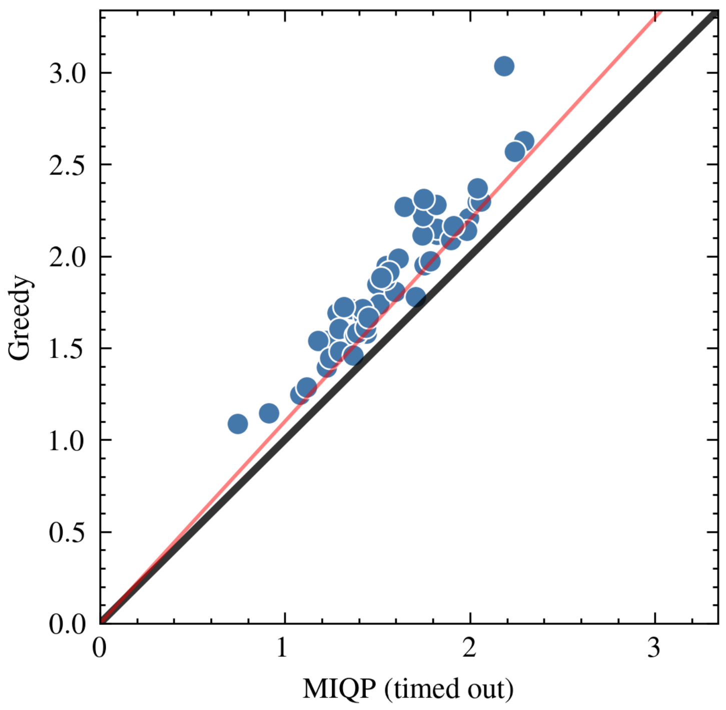


Figure 3.4: **Sensor Scheduling**: Comparison of solutions the MIQP (timed out after 10 seconds) and the greedy algorithm. The points above the black line indicate MIQP obtaining better solutions than greedy. The solution cost of the greedy approach of points above the red line are at least 10% worse than that of the MIQP solution cost.

Chapter 4

Informative Path Planning for Active Regression in Gaussian Processes

4.1 Introduction

The informative path planning (IPP) problem for active regression is a compelling challenge in robotics. Here, robots with limited resources collect measurements of an *unknown* function to produce an estimate with *minimum expected estimation error*. This is crucial in scenarios where data collection is expensive or hazardous. For instance, consider a team of autonomous underwater vehicles exploring the ocean to map temperature variations, a crucial task for marine ecosystem monitoring. The robots are equipped with temperature sensors and they aim to create an accurate temperature map. However, this operation is expensive in terms of energy and potentially risky for the robots. Apart from ocean monitoring [117, 38, 67, 83, 88, 119], this challenge also arises in other applications including mapping spatial fields [140, 122, 47], state estimation in linear dynamical systems [79, 111, 16], and sensor selection and scheduling [128, 20, 46].

The IPP problem considered in this chapter is known to be NP-hard since it contains the orienteering problem [54] as a special case. As a result, current approaches rely on heuristics to find sub-optimal solutions. However, we are interested in algorithms that compute *optimal* solutions: an avenue that has been largely unexplored for IPP. While these algorithms will have an exponential running time in the worst case, with the right design, could be very efficient on instances that arise in practice. The challenge lies in exploiting a suitable structure of the objective in active regression that would allow an algorithm to discard sub-optimal solutions quickly.

We model the target phenomena as a sample from a Gaussian Process (GP). This framework is widely used in IPP primarily due to three reasons. First, GPs provide an efficient way to compute a distribution over predictions with the mean serving as the point estimate and the variance quantifying the uncertainty. Second, the posterior variance at a test input depends *only* on the training and test data *inputs* and not the *measurements* (Remark 2.2). Hence, minimizing the variance, a typical goal in IPP, results in *deterministic* optimization problems. Third and crucial to our approach, the distribution mean is *linear* and *optimal* in the mean-squared sense and the distribution variance is *equal* to the expected squared estimation error, a result well-known in Bayesian estimation that has not been exploited in IPP. This property enables the formulation of IPP for active regression as a mixed integer program (MIP) which is the proposed approach in this chapter.

4.1.1 Contributions

First, we give the first exact formulation of IPP for active regression in GPs as a MIP. These MIPs are simple to implement in modern solvers and find provably optimal solutions. This approach has a desirable property that if the algorithm is terminated early, a solution is returned along with a sub-optimality certificate. Further, an additional benefit is typical routing constraints can be easily incorporated into the problem since our approach uses standard network flow techniques to model the paths. Second, we report computational results in simulation that demonstrate our approach can find optimal solutions in a variety of settings (large test sets, high connectivity graphs, multiple robots, large path budgets) and high quality sub-optimal solutions on large graphs across two popular kernels.

4.2 Problem Formulation

Given an environment $\mathcal{X} \subset \mathbb{R}^d$, the target phenomena to be estimated is modeled as an *unknown* function $f : \mathcal{X} \to \mathbb{R}$. We work in the Bayesian setting where f is a sample from a Gaussian Process with a zero mean function and covariance kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, i.e., $f \sim \mathrm{GP}(0, k)$.

Our goal is to plan paths of maximum utility for a multi-robot team on a graph defined on $\mathcal{X} \subset \mathbb{R}^d$. To this end, we consider the *pool-based* active learning setting where we have access to input data $\mathcal{V} := \{v_1, \ldots, v_n\} \in \mathcal{X}^n$ which forms the vertices of a directed weighted graph $G := (\mathcal{V}, \mathcal{E}, \ell)$ with $\ell(u, v)$ denoting the cost of edge (u, v). Note that we do not require ℓ to be a metric. Given a source vertex s and target vertex t, not necessarily distinct, an s-t path (open or closed) is a sequence of vertices $\langle u_1, \ldots, u_k \rangle$ with $u_1 = s$, $u_k = t$, and $(u_i, u_{i+1}) \in \mathcal{E}$ for $i = 1, \ldots, k-1$. For each vertex u_i on the path, the robot receives a measurement of the form

$$y_i = f(u_i) + \eta_i, \tag{4.1}$$

where $\eta_i \sim \mathcal{N}(0, \sigma^2)$ are zero-mean independently and identically distributed random variables.

We represent a path by a subset of edges $P \subseteq \mathcal{E}$ whose characteristic vector is the vector $\chi^P \in \{0,1\}^{|\mathcal{E}|}$ such that for any edge $e \in \mathcal{E}$,

$$\chi_e^P = \begin{cases} 1, & \text{if } e \in P \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

The set of all s-t paths in the graph is denoted by

$$\mathcal{P}_t^s := \{\chi^P \in \{0,1\}^{|\mathcal{E}|} : P \text{ is an } s\text{-}t \text{ path in } G\}. \tag{4.3}$$

With slight abuse of notation, we use P to refer to the path as well as the set of vertices visited on the path.

The objective we are interested in minimizing is the posterior variance evaluated on a given test set. For any weight matrix $M \succeq 0$ and a finite test set $T := \{t_1, \ldots, t_m\} \in \mathcal{X}^m$, define $\mathrm{PostVar}_M^T : 2^{\mathcal{V}} \to \mathbb{R}_{\geq 0}$ to be the M-weighted trace of the posterior variance evaluated on $T \in \mathcal{X}^m$ i.e.,

$$\mathrm{PostVar}_M^T(P) := \mathrm{trace}\left(M \bar{k}_{TT}^P\right), \tag{4.4}$$

where the posterior variance is given by (2.14):

$$\bar{k}_{TT}^P = \left(k_{TT} - k_{TP}\left(k_{PP} + \sigma^2 I_{|P|}\right)^{-1} k_{PT}\right) \in \mathbb{S}_+^{|T|} \tag{4.5}$$

and for any vertex subset $P = \{u_1, \ldots, u_k\} \subseteq \mathcal{V}$, we use

$$\begin{aligned} k_{TT} &= (k(t_i, t_j))_{1 \leq i,j \leq m} \in \mathbb{S}_+^m \\ k_{TP} = k'_{PT} &= (k(t_i, u_j))_{1 \leq i \leq m, 1 \leq j \leq k} \in \mathbb{R}^{m \times n} \\ k_{PP} &= (k(u_i, u_j))_{1 \leq i,j \leq k} \in \mathbb{S}_+^k. \end{aligned} \tag{4.6}$$

We are now ready to state the problem tackled in this chapter.

Problem 5 (IPP in GPs)**.** An instance of the IPP problem is the tuple $(G, k, T, r, A, b, s_1, \ldots, s_r, t_1, \ldots, t_r)$ where $G = (\mathcal{V}, \mathcal{E}, \ell)$ is a directed graph with vertex set $\mathcal{V} = \{v_1, \ldots, v_n\}$, k is the kernel of the underlying GP, $T := \{t_1, \ldots, t_m\}$ is the test set, $r \in \mathbb{N}$ is the number of robots, $A \in \mathbb{R}^{a \times r|\mathcal{E}|}$ and $b \in \mathbb{R}^a$ are the constraint matrix and vector, and finally, s_i and t_i are the start and target vertices for each $i \in [r]$. The goal is to compute paths $P_1, \ldots, P_r$ to

$$\begin{aligned} \text{minimize} \quad & \text{PostVar}_M^T \left(\cup_{i=1}^r P_i\right) \\ \text{subject to} \quad & A \begin{bmatrix} \chi^{P_1} \\ \vdots \\ \chi^{P_r} \end{bmatrix} \leq b, \\ & \chi^{P_i} \in \mathcal{P}_{t_i}^{s_i}, \quad i \in [r]. \end{aligned} \tag{4.7}$$

●

To summarize, we are looking for a collection of paths in a directed graph whose vertex locations minimize the posterior variance of the underlying GP on a finite set of test locations. We now discuss a few problem properties.

Remark (Polyhedral Constraints)**.** Representing the path as a characteristic vector allows for typical routing constraints for path planning to be captured by the linear inequality in (4.7). We give a few examples relevant to IPP below.

- *Path Length*: The well studied problem of a single robot with a path length constraint is modeled using a positive scalar budget b and a row vector $A \in \mathbb{R}^{|\mathcal{E}|}$ where for each $e = (u, v) \in \mathcal{E}$, $A_e = \ell(u, v)$. The extension to multiple robots with path budgets follows in a similar manner.

- *Sampling Costs*: Certain applications involve a sampling cost $c_v \in \mathbb{R}_{>0}$ for each vertex $v \in \mathcal{V}$. This is modeled by adding half the sampling cost of a vertex to all its incoming edges and outgoing edges. Specifically, the set of new edge costs is defined to be $\bar{\ell}(u, v) := \ell(u, v) + \frac{1}{2}c_u + \frac{1}{2}c_v$ (ignoring sampling costs at the start and end since those vertices will always be visited). Since any vertex on a path has exactly one incoming and one outgoing edge, this converts it into an equivalent path length constraint.

- *Expert Constraints*: In environmental monitoring, experts typically provide robot operating constraints. For example, robots may be allowed to collect at most a fixed number of data points in dangerous regions. Specifically, for any subset $V \subseteq \mathcal{V}$ representing a region of interest, one can enforce the sum of the number of outgoing edges for all vertices $v \in V$ to be at most a budget of $N > 0$.

Remark (Weighted Trace Objective)**.** The weighted trace (by a positive semidefinite matrix) is a generalization of the objectives typically considered in IPP. For example, it contains the well studied version of $M = I_m$ which equally weighs the error at all points in the test set. However, one can also target the cross covariances using the off-diagonal elements of M. In many situations, one may be interested in minimizing $\log\det(\bar{k}_{TT}^P)$. Unfortunately, it is concave in $\bar{k}_{TT}^P$ [14, Section 3.1.5] making the minimization difficult. However, one can find a local minimum via a series of weighted trace minimizations [48]. Let $\mathcal{C}$ represent the constraint set in (4.7). Using the idea in [48] yields the following iterative method:

$$\begin{aligned} P_{k+1} &= \min_{P\in\mathcal{C}} \operatorname{trace}\left(M_k \bar{k}_{TT}^P\right) \\ M_k &= \left(\bar{k}_{TT}^{P_k} + \delta I_m\right)^{-1}. \end{aligned} \tag{4.8}$$

Thus, minimizing the weighted trace can be used to find a locally optimal solution to the problem of minimizing $\log\det$.

4.3 Mixed Integer Convex Formulation

The challenge in optimally solving the IPP problem lies in identifying an appropriate structure of the set function (4.4) for optimization. While the decision variables for the edges have been set up in Problem 5, representing the objective in terms of these variables is not obvious. The main contribution of this chapter is the formulation of Problem 5 as a mixed integer program (MIP). We first present the key result that gives the MIP: viewing the weighted trace as a convex minimization problem for a given set of vertices. This is formalized below.

Theorem 5 (Objective Reformulation)**.** Define $c : \mathbb{R}^{m\times n} \to \mathbb{R}_{\geq 0}$ to be the function that maps the coefficients $K \in \mathbb{R}^{m\times n}$ of any linear estimator of $f \sim \mathrm{GP}(0,k)$ on the set $T \subset \mathcal{X}$ using the measurements $Y \in \mathbb{R}^n$ obtained at inputs $\mathcal{V} \subset \mathcal{X}$:

$$c(K) := \mathbb{E}\Big[(f_T - KY)' M (f_T - KY)\Big]. \tag{4.9}$$

Further, given a path P, let $y^P \in \{0,1\}^{|\mathcal{V}|}$ denote the characteristic vector of its visited vertices. Then, the objective in (4.4) can be written as the minimum of the following convex optimization problem:

$$\begin{aligned} \mathrm{PostVar}_M^T(P) = \min\Big\{ c(K) : K_{tv}(1 - y_v^P) = 0, \\ t \in T, v \in \mathcal{V} \Big\}. \end{aligned} \tag{4.10}$$

Theorem 5 is the key result that enables an exact formulation for Problem 5. It provides a convex formulation of the objective in terms of the characteristic vectors of the visited vertices making it amenable to mixed integer optimization. The proof is deferred to Section 4.3.1 since it requires intermediate results. The proof idea is to exploit the optimality of the posterior mean in Gaussian Process regression for functions of the squared loss (Section 4.3.1). The last remaining piece is to connect the edge characteristic vectors in Problem 5 to the vertex characteristic vectors which is accomplished using standard network flow techniques (Section 4.3.2).

4.3.1 Quadratic Formulation

Our roadmap to the quadratic formulation is as follows. First, we view the GP posterior mean as the linear least squares estimator (LLSE) and show that the posterior variance is equal to the mean squared estimation error. While this is known in Bayesian estimation [34, Chapter 3], it has not been exploited for IPP. Next, we show that the LLSE is optimal for a class of functions of the mean squared error including the weighted trace in Problem 5 finally leading to the proof of Theorem 5.

GP Regression and MMSE We begin with an alternative perspective on the Gaussian Process predictive equations (2.14). Typically, the equations are derived using Gaussian conditioning identities [73, Theorem 3.1]. However, the view through the lens of minimum mean squared error estimation (MMSE) allows us to formulate the IPP problem for active regression *exactly*. The following result is also known as the *best linear unbiased predictor* property of the posterior mean. The proof is relatively straightforward and we include it for completeness.

Theorem 6 (GP Regression Optimality)**.** Assume the setup governed by equations (2.10) and (2.11). Let $X \in \mathcal{X}^n$ and $Y \in \mathbb{R}^n$ denote a set of inputs and measurements respectively. In addition, let $T \in \mathcal{X}^m$ denote the test set. Then, the posterior mean $\bar{m}_T^X := \mathbb{E}[f_T|Y] \in \mathbb{R}^m$ in (2.14) is the *linear least squares estimator* of f_T given measurements Y i.e.,

$$\bar{m}_T^X = K_* Y, \tag{4.11}$$

where $K_* \in \mathbb{R}^{m \times n}$ is computed by minimizing the expected squared error over the class of linear estimators i.e.,

$$\begin{aligned} K_* &:= \arg\min_K \mathbb{E}\left[(f_T - KY)'(f_T - KY)\right] \\ &= k_{TX}\left(k_{XX} + \sigma^2 I_n\right)^{-1}. \end{aligned} \tag{4.12}$$

Further, the posterior covariance $\bar{k}_{TT}^X \in \mathbb{S}_+^m$ is equal to the error covariance resulting from the posterior mean i.e.,

$$\begin{aligned}\bar{k}_{TT}^X &= \mathbb{E}\left[\left(f_T - \bar{m}_T^X\right)\left(f_T - \bar{m}_T^X\right)'\right] \\ &= \mathbb{E}\left[\left(f_T - K_*Y\right)\left(f_T - KY\right)'\right].\end{aligned} \tag{4.13}$$

Proof. The proof follows from an application of Theorems 1 and 2. The measurements $Y \in \mathbb{R}^n$ and GP function values $f_T \in \mathbb{R}^m$ are jointly Gaussian random vectors such that

$$\begin{bmatrix} f_T \\ Y \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} k_{TT} & k_{TX} \\ k_{XT} & k_{XX} + \sigma^2 I_n \end{bmatrix}\right). \tag{4.14}$$

By the application of Theorem 2 with $x = f_T$ and $y = Y$, the MMSE estimator i.e., the posterior mean $\mathbb{E}\left[f_T|Y\right]$, is linear in Y. In the notation of Theorem 1, this corresponds to $\Sigma_{xy} = k_{TX}$, $\Sigma_{xx} = k_{TT}$, and $\Sigma_{yy} = k_{XX} + \sigma^2 I_n$. Now, the application of Theorem 1 gives (4.11) and (4.12). Finally, the posterior covariance $\bar{k}_{TT}^X$ is *not* a random variable (Remark 2.2) and using the definition of the posterior variance in (2.14),

$$\begin{aligned}\bar{k}_{TT}^X = \mathbb{E}\Big[\bar{k}_{TT}^X\Big] &= \mathbb{E}\Big[\mathbb{E}\left[(f_T - \bar{m}_T^X)(f_T - \bar{m}_T^X)'|Y\right]\Big], \\ &= \mathbb{E}\left[(f_T - \bar{m}_T^X)(f_T - \bar{m}_T^X)'\right].\end{aligned} \tag{4.15}$$

□

The significance of Theorem 6 is that the GP posterior mean is the MMSE estimator and is *linear*. In the following section, we will focus on the optimality properties of linear estimators for a class of objective functions including the weighted trace.

LLSE Optimality The coefficients of the optimal linear estimator minimizes the expected squared error as shown in Theorem 1. We will show that these coefficients are also the minimizers of the class of matrix non-decreasing functions of the posterior variance (equivalently, the error covariance by Theorem 6). First, we define a matrix non-decreasing function.

Definition 4 (Matrix Non-Decreasing Function)**.** A function $g : \mathbb{S}^n \to \mathbb{R}$ is called matrix non-decreasing if it is monotone with respect to the positive semidefinite cone i.e.,

$$x \succeq y \implies g(x) \geq g(y). \tag{4.16}$$

Lemma 5 (Optimality for Matrix Non-Decreasing Functions)**.** Let x and y be jointly distributed random vectors (of sizes m and n) with zero mean and covariance given by

$$\begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \in \mathbb{S}_+^{m+n}. \tag{4.17}$$

Denote $h : \mathbb{R}^{m\times n} \to \mathbb{S}_+^m$ to be the function that maps the coefficients $K \in \mathbb{R}^{m\times n}$ of any linear estimator of x given y, to its associated error covariance i.e.,

$$h(K) := \mathbb{E}\left[(x - Ky)(x - Ky)'\right]. \tag{4.18}$$

Let $g : \mathbb{S}_+^m \to \mathbb{R}_{\geq 0}$ be any non-negative matrix non-decreasing function and define $c : \mathbb{R}^{m\times n} \to \mathbb{R}_{\geq 0}$ to be the composition

$$c = g \circ h. \tag{4.19}$$

Then, the LLSE coefficients $K_* = \Sigma_{xy}\Sigma_{yy}^{-1} \in \mathbb{R}^{m\times n}$ satisfy

$$K_* \in \underset{K\in\mathbb{R}^{m\times n}}{\arg\min}\; c(K). \tag{4.20}$$

Proof. We will show that for any coefficients $K \in \mathbb{R}^{m\times n}$,

$$c(K) \geq c(K_*). \tag{4.21}$$

This follows from a combination of Lemma 2 and Definition 4. Lemma 2 states that for any $K \in \mathbb{R}^{m\times n}$,

$$h(K) \succeq h(K_*). \tag{4.22}$$

Then, using the fact that g is matrix non-decreasing gives

$$\begin{aligned} g(h(K)) &\geq g(h(K_*)) \\ \implies c(K) &\geq c(K_*). \end{aligned} \tag{4.23}$$

$\square$

The key takeaway of Lemma 5 is we can recover the LLSE coefficients by optimizing over the class of functions that are compositions of matrix non-decreasing functions of the error covariance matrix i.e., the functions given by c in (4.19) over the set of linear estimators described by the coefficients K. This is a critical property that will help represent the objective as a convex minimization problem. This result is important because several objectives of interest for IPP are matrix non-decreasing functions [14, Example 3.46] on $\mathbb{S}_+^m$:

- *Weighted Matrix Trace*: $g(X) = \operatorname{trace}(MX)$.
- *Log Determinant*: $g(X) = \log\det(X)$.

Mixed Integer Formulation We now have the required results to prove Theorem 5. Recall for a path P, $y^P \in \{0,1\}^{|\mathcal{V}|}$ is the characteristic vector of the vertices visited on the path i.e., for any vertex $v \in \mathcal{V}$,

$$y_v^P = \begin{cases} 1, & \text{if } v \in P \\ 0, & \text{otherwise.} \end{cases} \tag{4.24}$$

The idea of the proof is to optimize over the class of linear estimators $K \in \mathbb{R}^{m\times n}$ using measurements at *all* n vertices with the constraint that the coefficient corresponding to a vertex can be used only if the vertex is visited on the path. We now present the proof which will make this idea precise.

Proof of Theorem 5. Consider the constraint $K_{tv}(1-y_v^P) = 0$. It gives the following equivalence: $y_v^P = 0 \implies K_{tv} = 0$ for all $t \in T$ i.e., the measurement at vertex v cannot be used in the estimator since it was not visited on the path P or equivalently, the column in K corresponding to the vertex v is zero. Thus, when taking a linear combination KY, the measurements in Y corresponding to the zero columns can be dropped. If we denote $Y_P \in \mathbb{R}^{|P|}$ as the subset of measurements corresponding to the visited vertices P, then we can rewrite the minimization as

$$\begin{aligned} &\min\Big\{c(K) : K_{tv}(1-y_v^P) = 0,\ t \in T, v \in \mathcal{V}\Big\}, \\ &= \min_{K\in\mathbb{R}^{m\times|P|}} \mathbb{E}\Big[(f_T - KY_P)'M(f_T - KY_P)\Big], \\ &= \min_{K\in\mathbb{R}^{m\times|P|}} \operatorname{trace}\Big(M\mathbb{E}\Big[(f_T - KY_P)(f_T - KY_P)'\Big]\Big), \end{aligned} \tag{4.25}$$

where the last line follows from the linearity of the trace and expectation and the fact that $\operatorname{trace}(AB) = \operatorname{trace}(BA)$ for any conformable matrices A, B. The last step in the proof is an appropriate application of Lemma 2. Let $x = f_T \in \mathbb{R}^m$, $y = Y_P \in \mathbb{R}^{|P|}$. Then, the function $h : \mathbb{R}^{m\times|P|} \to \mathbb{S}_+^m$ in Lemma 5 is given by

$$\begin{aligned} h(K) &= \mathbb{E}\left[(f_T - KY_P)(f_T - KY_P)'\right] \\ &= K(k_{PP} + \sigma^2 I_{|P|})K' - k_{TP}K' - Kk_{PT} + k_{TT}. \end{aligned} \tag{4.26}$$

Now, let $g : \mathbb{S}_+^m \to \mathbb{R}^{\geq 0}$ be the weighted matrix trace

$$g(X) := \operatorname{trace}(MX). \tag{4.27}$$

Then, applying Lemma 5 gives the optimal solution to (4.25) as $K_* = \Sigma_{xy}\Sigma_{yy}^{-1} = k_{TP}(k_{PP} + \sigma^2 I_{|P|})^{-1}$ with value

$$\begin{aligned}
&\min_{K \in \mathbb{R}^{m \times |P|}} \operatorname{trace}\Big(M\mathbb{E}\Big[(f_T - KY_P)(f_T - KY_P)'\Big]\Big) \\
&= \operatorname{trace}\Big(M\mathbb{E}\Big[(f_T - K_*Y_P)(f_T - K_*Y_P)'\Big]\Big) \\
&= \operatorname{trace}(M\bar{k}_{TT}^P) = \operatorname{PostVar}_M^T(P),
\end{aligned} \tag{4.28}$$

where the last equality follows from an application of Theorem 6 with $X = P$ and $Y = Y_P$.

Finally, we will show the minimization problem in the RHS of (4.10) is convex by showing c is convex and the equality constraints are affine. It is clear that for any fixed path P, the constraint $K_{tv}(1 - y_v^P) = 0$ is affine in K. Notice that $c = g \circ h$ is a composition of g and h. Using a composition theorem for convexity [14, Section 3.6.2], we have that c is convex since g is matrix non-decreasing (since $M \succeq 0$) [14, Example 3.46] and h is convex [14, Example 3.49]. □

4.3.2 Network Flow Formulation

The IPP problem for active regression in Gaussian Processes (4.7) is formulated as the following mixed integer program:

$$\begin{aligned}
&\text{minimize} \quad \operatorname{tr}\left(M\left(K(k_{\mathcal{V}\mathcal{V}}+\sigma^2 I_n)K' - 2k_{T\mathcal{V}}K' + k_{TT}\right)\right)\\
&\text{subject to}\\
&\quad \sum_{e\in\mathcal{E}_v^{\text{in}}} \chi_e^i = \sum_{e\in\mathcal{E}_v^{\text{out}}} \chi_e^i \le 1, \quad i\in[r], v\in\mathcal{V}\setminus\{s_i,t_i\}, && (4.29)\\
&\quad \sum_{e\in\mathcal{E}_{t_i}^{\text{in}}} \chi_e^i = \sum_{e\in\mathcal{E}_{s_i}^{\text{out}}} \chi_e^i = 1, \quad i\in[r], && (4.30)\\
&\quad \sum_{e\in\mathcal{E}_{s_i}^{\text{in}}} \chi_e^i = \sum_{e\in\mathcal{E}_{t_i}^{\text{out}}} \chi_e^i = 0, \quad i\in[r], && (4.31)\\
&\quad \sum_{\substack{e=(u,v)\in\mathcal{E}:\\ u,v\in S}} \chi_e^i \le |S|-1, \quad i\in[r], S\subset\mathcal{V}\setminus\{s_i,t_i\}, && (4.32)\\
&\quad y_v \ge \sum_{e\in\mathcal{E}_v^{\text{in}}} \chi_e^i, \quad i\in[r], v\in\mathcal{V}\setminus\cup_{i=1}^r\{s_i,t_i\}, && (4.33)\\
&\quad y_v \le \sum_{i=1}^r \sum_{e\in\mathcal{E}_v^{\text{in}}} \chi_e^i, \quad v\in\mathcal{V}\setminus\cup_{i=1}^r\{s_i,t_i\} && (4.34)\\
&\quad K_{tv}(1-y_v) = 0, \quad t\in T, v\in\mathcal{V}\setminus\cup_{i=1}^r\{s_i,t_i\}, && (4.35)\\
&\quad A\begin{bmatrix}\chi^1\\ \vdots\\ \chi^r\end{bmatrix} \le b, \quad i\in[r], && (4.36)\\
&\quad \chi^i\in\{0,1\}^{|\mathcal{E}|}, \quad i\in[r], && (4.37)\\
&\quad y\in\{0,1\}^{|\mathcal{V}|}, && (4.38)\\
&\quad K\in\mathbb{R}^{m\times n}, && (4.39)
\end{aligned}$$

where the covariance matrices across the test set $T = \{t_1,\dots,t_m\}$ and vertices $\mathcal{V} = \{v_1,\dots,v_n\}$ are given by the kernel of the underlying Gaussian Process $f \sim \mathrm{GP}(0,k)$ i.e.,

$$\begin{aligned}
k_{TT} &= (k(t_i,t_j))_{1\le i,j\le m} \in \mathbb{S}_+^m\\
k_{T\mathcal{V}} &= (k(t_i,v_j))_{1\le i\le m, 1\le j\le n} \in \mathbb{R}^{m\times n}\\
k_{\mathcal{V}\mathcal{V}} &= (k(v_i,v_j))_{1\le i,j\le n} \in \mathbb{S}_+^n.
\end{aligned} \qquad (4.40)$$

We start by introducing the decision variables $\chi^i \in \{0,1\}^{|\mathcal{E}|}$ which represent the path of robot i in the graph where $\chi^i_e = 1$ if edge e is traversed by robot i and 0 otherwise. The sets $\mathcal{E}^{\text{in}}_v := \{(u,v) \in \mathcal{E}\}$ and $\mathcal{E}^{\text{out}}_v := \{(v,u) \in \mathcal{E}\}$ represent the set of incoming and outgoing edges from vertex v. The constraints (4.29), (4.30), (4.31), (4.32) enforce the s-t path constraint for all robots using network flow techniques. Specifically, constraints (4.29) ensure that all vertices (except the start and target) have the in-degree equal to the out-degree which can be at most equal to 1 (if the vertex is visited). Then, (4.30) and (4.31) enforce the in-degree and out-degree constraints for the start and end nodes. The constraints in (4.32) prevent ensure no subtours are allowed i.e., on any subset of vertices of size k, the number of edges with both endpoints in that subset must be less than or equal to $k-1$ (otherwise it would contain a cycle). We then introduce the decision variables $y \in \{0,1\}^{|\mathcal{V}|}$ to represent whether vertex v is visited (by at least one robot). Constraints (4.33) and (4.34) connect the edge decision variables to the vertex decision variables. Specifically, (4.33) ensures for every vertex v, if a robot i visits it i.e., there is an incoming edge to v, then y_v must be set to 1. Next, (4.34) ensures that if no robots visit a vertex v, y_v must be set to 0. Finally, the objective and constraint (4.35) follow from an application of Theorem 5.

The number of decision variables is $r|\mathcal{E}| + |\mathcal{V}| + mn$ which is $O(rn^2 + mn)$ where m is the number of test points and n is the number of graph vertices. The constraint (4.35) can be implemented as an SOS Type 1 constraint [9]. However, the constraint set described by (4.32), which we refer to as the *subtour constraints*, is exponential in the number of vertices preventing its enumeration at the beginning. There are two standard techniques to deal with this issue: the Miller-Tucker-Zemlin (MTZ) formulation [93] or a lazy callback approach [91]. We evaluate both techniques in simulation.

MTZ Formulation This introduces extra variables to eliminate subtours and replaces (4.32) with new constraints. Specifically, for each $i \in [r]$ and each vertex $v \in \mathcal{V}$, we introduce the variables $a^i_v \in \mathbb{R}$ and the constraints

$$\begin{aligned} a^i_{s_i} &= 1, \quad && i \in [r] \\ 2 \le a^i_v &\le n, \quad && v \in \mathcal{V} \setminus \{s_i\}, i \in [r] \\ a^i_u - a^i_v + 1 &\le (n-1)(1-\chi^i_e), && e = (u,v) \in \mathcal{E}, i \in [r]. \end{aligned} \tag{4.41}$$

This is a well-known method for eliminating subtours and the reader is referred to [93] or [103] for more details.

Lazy Callback The alternative to the MTZ formulation is a lazy approach that follows the algorithm introduced in [91]. First, the MIP is solved *without* the subtour constraints (4.32). When an integer solution is found, it is tested for feasibility i.e., does the solution contain any subtours? If the solution is feasible, it is optimal and the algorithm terminates. If not, the violated subtours of the form (4.32) are added to the constraint set and the resulting MIP is solved again. Note that at each iteration, finding subtours in the current solution is easy: it is sufficient to find a connected component in the subgraph given by the edges of the solution which does not include the start and end nodes $s, t \in \mathcal{V}$. In the worst case, the algorithm would enumerate *all* the subtour inequality constraints described by (4.32), suggesting that this approach is impractical. Further, one needs to solve a MIP at each iteration, which in the worst case, requires an exponential number of leaves to explore in the branch and bound algorithm. However, in practice (Section 4.4), one only requires a few inequalities to be added before arriving at an optimal solution. Note that this approach solves a sequence of MIPs, each differing only by a few linear inequalities (the violated subtour constraints). A naive implementation would involve the creation of a new branch and bound tree (to solve the MIP) each time a new set of inequalities are added, which is computationally inefficient. We implement a single tree solution using dynamic constraint generation, known in the optimization literature as a lazy constraint or a column generation method. Lazy constraints add constraints whenever an integer solution is found [4], saving the trouble of recreating the search tree each time the MIP needs to be resolved. This is straightforward to implement in modern solvers including Gurobi [60] and CPLEX [32].

4.3.3 Efficient Warm Starts for Budgeted Paths

In this section, we provide the details of the greedy algorithm used to warm start our MIP in the setting of path length constraints. This improves the computational speed in practice and in general, it is better to start the search with a high quality feasible solution yielding a good upper bound which helps prune large parts of the search tree. The greedy algorithm has strong empirical results when minimizing the trace (without the path constraints) and is a natural choice.

The greedy algorithm for the point-to-point IPP problem differs from its rooted counterpart. In the rooted version, the next point that maximizes the marginal gain (normalized by the cost of the edge) is selected until the robot runs out of budget. However, this idea does not work in the point-to-point version since the robot must end up at the target node. We adapt the idea from [52] for the single robot case (Algorithm 1). Here, the algorithm finds an initial feasible path (Line 1) from start to target (for example, the shortest path) and greedily adds vertices to the tour that maximize the error reduction (Line 8). To minimize the resulting path lengths, we use the 2-OPT heuristic (Line 7). The extension to the multi-robot case follows the idea proposed in [13], where one plans multiple paths sequentially using the single robot greedy algorithm as a subroutine. Specifically, at iteration i, the path for robot i is planned using the single robot greedy algorithm where the vertices visited by robots $1, \dots, i-1$ are added to the collected measurement set for robot i (so that there is no benefit for robot i to visit a previously visited vertex).

Algorithm 1: SingleRobotGreedy

```
Input: Graph $G = (\mathcal{V}, \mathcal{E}, \ell)$, budget $B$, start node $s$, target node $t$, visited
       locations $S \subset \mathcal{V}$
Output: $s$-$t$ path $P$
1  $P^* \leftarrow$ InitialFeasiblePath($s$, $t$)
2  do
3      $P \leftarrow P^*$
4      best $\leftarrow 0$
5      $P^* \leftarrow null$
6      for $v \in \mathcal{V}$ do
7          $P' \leftarrow$ 2-OPT$(P \cup S \cup \{v\})$
8          margin $\leftarrow \frac{\text{PostVar}^T_M(P \cup S) - \text{PostVar}^T_M(P' \cup S)}{\text{Cost}(P') - \text{Cost}(P)}$
9          if $margin > best$ and $Cost(P') \leq B$ then
10             best $\leftarrow$ margin
11             $P^* \leftarrow P'$
12 while $P^* \neq null$
13 return $P$
```

4.4 Numerical Results

We demonstrate the effectiveness of our approach on several numerical experiments. First, we discuss how our approach can be used in real world settings using an illustrative example (Section 4.4.1). Second, we present an analysis on randomly generated instances to benchmark the MIP against the commonly used greedy algorithm (Section 4.4.2). All experiments are run using Gurobi 11.0 (free for academic use) on a computer with 16GB RAM and an AMD Ryzen 7 processor.

4.4.1 Illustrative Example: Elevation Mapping on Mt. St. Helens

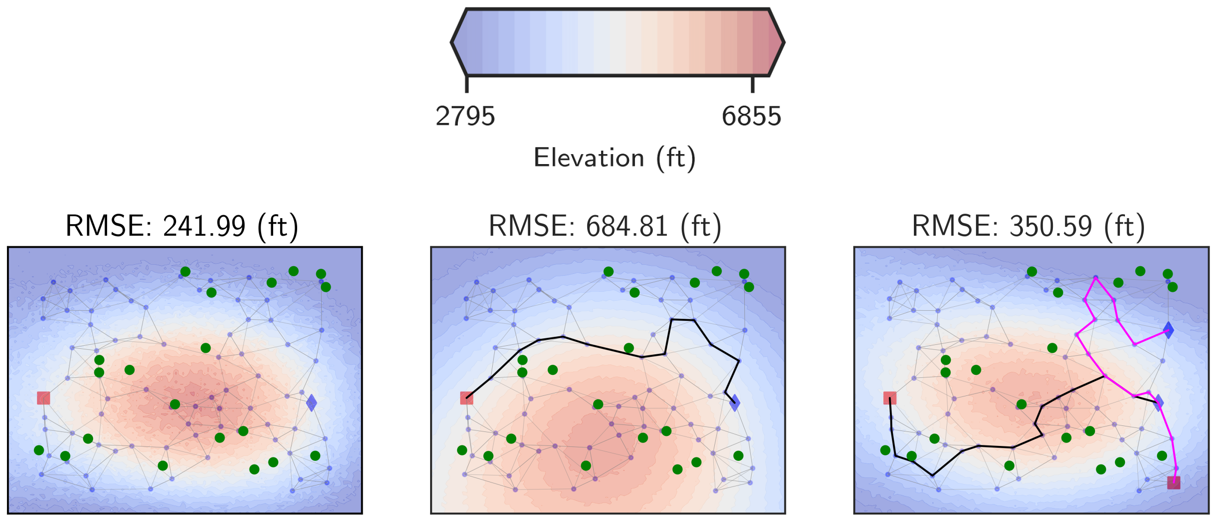
We begin with a qualitative example on an elevation mapping dataset for Mount St. Helens in Washington, USA. Robot terrain mapping using GPs have been considered before [130] and the preprocessed data is publicly available [1] [25].

The input data consists of 2D coordinates and the corresponding measurements are the elevation values. We select a subset of the data to perform GP model selection. Specifically, we consider the squared exponential kernel and estimate its hyperparameters by maximizing the marginal likelihood of the data (see [135, Chapter 5] for details). The graph vertices and test set are selected to be a subset of the training input data and test input data respectively. Each vertex is connected to its four nearest neighbours with the edge weights given by the Euclidean distance. Figure 4.1a shows the graph (blue vertices, black edges) and the test set (green circles). The robot plans a path from start (red square) to target (blue diamond) to minimize the expected estimation error on the test set.

The plot in Figure 4.1a shows the estimated elevation map as well as the *empirical root mean squared error* (RMSE) at the test locations if the robot had collected measurements at *all* the vertices. The RMSE is the average squared error computed using the *actual dataset measurements* versus the expected squared error in the objective function for IPP. Under the specified GP model, this estimate (contour plot in Figure 4.1a), with an RMSE of 242 feet, is the best any feasible solution to the IPP problem can hope to achieve. A typical solution to IPP will not visit all graph vertices due to resource constraints resulting in higher RMSE as we will see below.

Figure 4.1b shows the optimal planned path for a single robot under a path length constraint. The plot shows the estimated elevation map resulting from the measurements collected on the path vertices, which achieves an RMSE of 685 feet. Due to the budget constraint, the RMSE is nearly 2.8 times worse than if the robot visited all locations. Note that visiting the top-right region in Figure 4.1b would yield a significant reduction in error due to the presence of multiple test points (green circles). However, the robot is unable to visit that region and yet, the estimated map in that region seems visually accurate when compared to Figure 4.1a. This is because the variables in this GP are highly correlated. Thus, measurements give reasonable information about points that are further away and since the robot takes measurements that are somewhat close to test points in this region, the estimates are moderately accurate. On the other hand, if the closest measurement to a test location is very far away, then there is no hope for a good estimate. This is seen

[1] github.com/Weizhe-Chen/attentive_kernels



(a) The plot shows the RMSE and estimated map using measurements at *all* vertices. This is the best any IPP feasible solution can achieve.

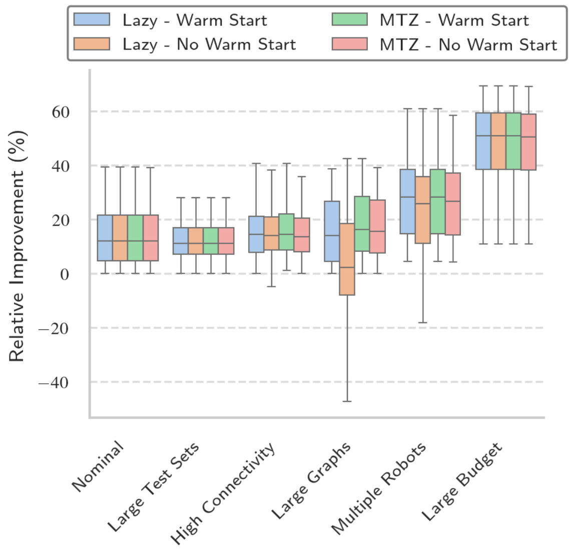
(b) The optimal single robot path. The robot cannot visit all vertices due to budget constraints and the resulting map estimate is inaccurate.

(c) The planned paths for two budget constrained robots. The addition of another robot improves the estimation quality significantly.

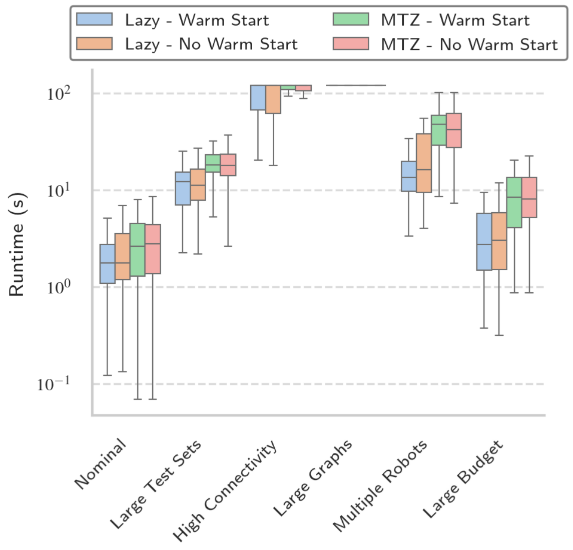
Figure 4.1: Elevation mapping on Mount St. Helens. The robots plan budgeted paths from start (red square) to target (blue diamond) on the graph vertices (blue circles) with the goal of minimizing the expected estimation error on the test points (green circles).

in the lower half region of the plot in Figure 4.1b which is unexplored by the robot. As a result, the robot predicts the elevation values to be close to the maximum value whereas the best estimates are closer to the minimum as seen in Figure 4.1a.

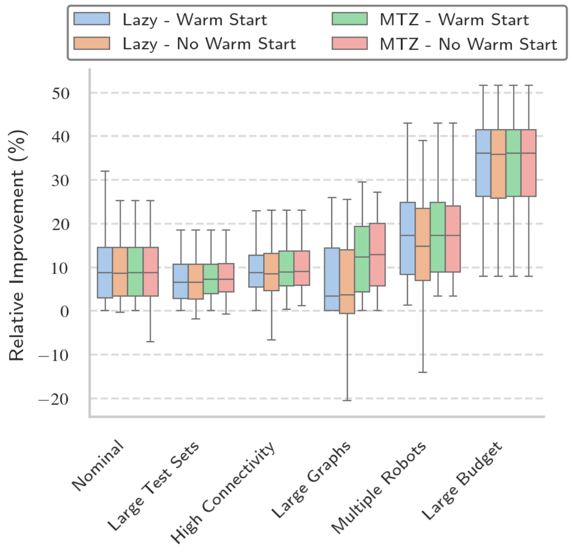
Our approach also handles multiple robots planning budgeted paths with distinct start and target vertices as shown in Figure 4.1c. The addition of a second robot allows for higher exploration resulting in a more accurate map when compared to the single robot setting in Figure 4.1b as evidenced by the RMSE of 351 feet. By visiting less than half the number of graph vertices, the multi-robot achieves an error 1.45 times worse than if it had collected all the data as in Figure 4.1a. Here, we see that the paths are significantly different from the single robot case. Thus, even if one has a near-optimal single robot planner, it may not be sufficient to generate near-optimal plans in the multi-robot setting, motivating the need to plan jointly for multiple robots such as the formulated MIP.



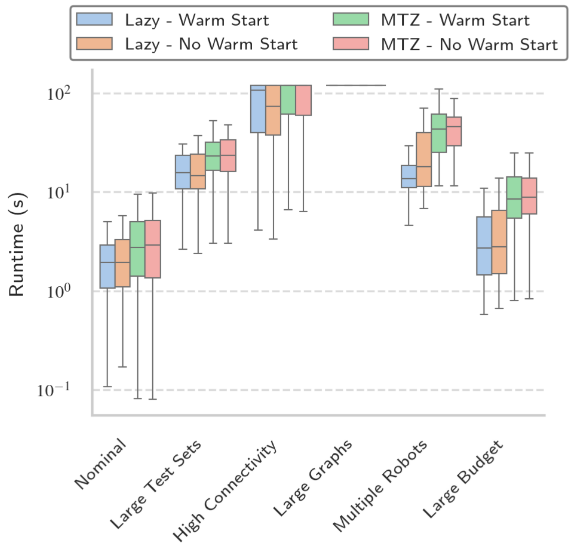
(a) MIP Performance - Squared Exponential Kernel



(b) MIP Runtime - Squared Exponential Kernel



(c) MIP Performance - Matern Kernel



(d) MIP Runtime - Matern Kernel

Figure 4.2: The analysis of MIP solution quality and runtime over different batch settings for two kernels: squared exponential and Matern. We run each algorithm on 50 different instances per batch setting and record the solution and the runtime. (a) and (c) show the relative improvement in MIP solution quality over the greedy approach. (b) and (d) show the MIP runtime (log scale) across different batch settings.

4.4.2 Analysis of Random Instances

In this section, we provide numerical results showing that i) optimally solving the proposed MIP yields significant improvements in solution quality over a greedy baseline, and ii) we can achieve a practical runtime for moderate problem instances. To this end, we present a statistical analysis of algorithm performance on the well-studied problem of planning budgeted paths. The baseline algorithm we compare against is the greedy approach described in Section 4.3.3, which is popular in orienteering and IPP. As discussed in Section 4.3.2, there are two ways to implement the subtour constraints: the MTZ formulation and the lazy callbacks. Further, we also investigate the role of warm starting the MIP using the greedy solutions. Thus, we evaluate the performance of the following four approaches: `Lazy - Warm`, `Lazy - No Warm`, `MTZ - Warm`, and `MTZ - No Warm`. As the names suggest, these implement the lazy callbacks and MTZ methods for subtour elimination with and without the greedy warm starts. We will see that `MTZ - Warm` is the preferred approach, both in terms of solution quality and runtime, on larger instances whereas `Lazy - Warm` finds optimal solutions on smaller instances more quickly. We measure the performance of each algorithm by the expected estimation error and the cost of an algorithm by its runtime in seconds. We set a timeout of 120 seconds and return the best solution found in that time frame.

Instance Setup: The instances for the budgeted path problem are constructed as follows. The environment we consider is a 50×50 square. We sample m points uniformly at random in the square to generate the test set T. We construct a probabilistic roadmap with n vertices, connection factor k, and weights given by the Euclidean distance. The robots must traverse this graph with path length budget B. We first solve 50 instances on a nominal set of parameters: number of graph vertices $n = 40$, number of test points $m = 20$, connection factor $k = 3$, budget $B = 100$, and number of robots $r = 1$. Then, we solve five other batches of problems where in each batch, a different set of parameters is increased by a factor of 3. Note that we run 50 instances for each batch setting. These batches are large graphs (n from 40 to 120), high connectivity (k from 3 to 9), large budgets (B from 100 to 300), large test sets (m from 20 to 60), and multiple robots (r from 1 to 3).

We run our simulations using two widely used kernels in GP regression [135]: the squared exponential with lengthscale $L = 5$, signal standard deviation $\sigma_0 = 100$ and the Matern 3/2 kernel with lengthscale $L = 5$. Further, we assume the measurements are corrupted with zero mean noise and standard deviation $\sigma = 0.1$. Note that the parameter L roughly measures how quickly the function changes as the distance increases i.e., as L increases, the function values at points further apart are correlated. To give some intuition for these parameters choices, one would have to sample at a distance of 4.13 units from a test point to reduce the associated expected squared estimation error by $\approx 50\%$ when using the squared exponential kernel in the environment described above. Note that if the length scale parameter L is close to zero, measurements would yield very little information about the function values at the test points which is not an interesting setup. On the other hand, if L is very large, most feasible paths yield good solutions since a few measurements will suffice to give good estimation quality. As a result, we select an approximate middle ground where L is neither too high nor too low. Finally, the parameter σ_0^2 is a scaling factor and the measurement noise variance σ^2 impacts the reliability of each measurement: higher variance implies a noisier measurement which in turn implies lower reduction in expected estimation error.

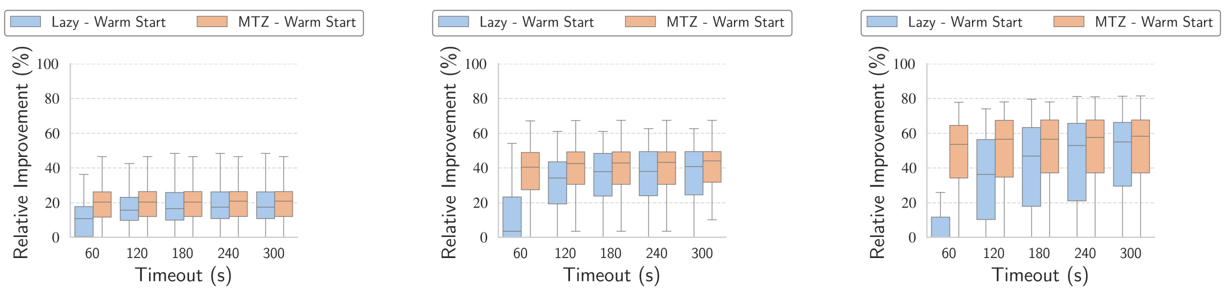
We plot the *relative performance* with respect to the greedy algorithm as well as the *absolute* runtimes for both kernels in Figure 4.2. We discuss the following observations.

Improved Solution Quality over Greedy In Figure 4.2a, we observe significant improvements in solution quality using the MIP over the greedy algorithm for the squared exponential kernel. Specifically, the best performing MIP exhibits the following approximate median improvements: 12% for nominal (`Lazy - Warm`), 11.1% (`Lazy - Warm`) for large test sets, 14.6% (`MTZ - Warm`) for high connectivity, 16.3% (`MTZ - Warm`) for large graphs, 28.2% (`Lazy - Warm`) for multiple robots, and 50.8% (`Lazy - Warm`) for large budgets. In the case of the Matern kernel (Figure 4.2c), the median improvements are: 8.7% for nominal (`Lazy - Warm`), 7.2% for large test sets (`MTZ - Warm`), 9% for high connectivity (`MTZ - Warm`), 12.8% for large graphs (`MTZ - Warm`), 17.2% for multiple robots (`MTZ - Warm`), and 36% for large budgets (`MTZ - Warm`). The sub-optimality of the greedy algorithm for single and multi-robot planning is apparent in the batches of multiple robots and large budgets. The drastic performance improvement in large budgets is interesting. When the budget is low $B = 100$, the optimal path is not very different from the shortest path between the start and target nodes (which is used to provide an initial feasible solution to greedy). This is the reason for the minor improvements in the batches apart from large budgets. However, for larger budgets, the optimal paths are quite different from the shortest path, leading to much lower expected estimation errors. Of course, one could use different initial feasible solutions for greedy but it is not clear which scheme would yield the best performance. Finally, `MTZ - Warm` is preferred slightly over `Lazy - Warm` due to the gap in solution quality on large graphs for Matern kernels (Figure 4.2c) and similar performance elsewhere.

Importance of Warm Starts We find that warm starts are crucial to finding high quality solutions, especially in the large graph and multiple robot setting. In Figures 4.2a and 4.2c, we see multiple instances of negative relative improvement: `Lazy - No Warm` on high connectivity, large graphs, and multiple robots in Figure 4.2a, `MTZ - No Warm` on nominal and large test sets (in addition to high connectivity, large graphs, and multiple robots) in Figure 4.2c. On these instances, a negative relative improvement implies the best returned solutions were of *worse* quality than greedy. In contrast, warm starting enables the solver to find better quality solutions as evidenced by the improvement on large graphs in Figure 4.2a and on multiple robots in both Figures 4.2a and 4.2c.

Efficient Runtimes We have established that computing solutions with the MIP leads to a substantial improvement in solution quality over the greedy algorithm. This naturally leads to the question: how long does it take to find these optimal/sub-optimal solutions? Note that the greedy algorithm computes solutions in under a second making it *extremely* efficient. In Figure 4.2b, we observe that the median runtimes (in seconds) of `Lazy - Warm` are as follows: 1.8 (nominal), 12.2 (large test sets), 120 (high connectivity, large graphs), 13.6 (multiple robots), and 2.8 (large budgets). For the Matern kernel (Figure 4.2d), the median runtimes are: 1.9 (nominal), 15.7 (large test sets), 108 (high connectivity), 120 (large graphs), 13.6 (multiple robots), and 2.7 (large budgets). These results are encouraging since the majority of the instances are solved to optimality (recall the timeout is 120 seconds). Further, `Lazy - Warm` has faster runtimes compared to `MTZ - Warm` especially in the case of multiple robots and large budget. The challenging instances are on high connectivity and large graphs where the median runtimes are 120 seconds indicating time outs. This does not come as a surprise as the number of MIP decision variables scales linearly with the number of edges as well as the number of vertices increasing the time required to find optimal solutions. In any case, we dig deeper and investigate the relationship between solution quality and runtime below on large graphs below.

Solution Quality vs Runtime We study the quality of returned MIP solutions (in terms of relative improvement over greedy) as a function of its runtime on large graphs. This setting was particularly challenging for the MIP when the timeouts were set to 120 seconds in the analysis above. We aim to understand if increasing the timeout will yield significantly higher quality solutions over the greedy algorithm. On each generated problem instance, the MIPs are progressively timed out at 60 second intervals and the returned solutions at each interval are recorded. We consider three budget settings: low ($B = 100$), moderate ($B = 150$), and high ($B = 200$). We generate 50 instances for each budget with the other parameters matching the large graph batch setting (as described at the



(a) Low Budget (b) Moderate Budget (c) Large Budget

Figure 4.3: The relative performance of the MIP on large graphs with varying budgets (low, moderate, large). Each plot shows the statistics of the relative improvement in solution quality over greedy as a function of the MIP timeout (in seconds).

beginning of this section) and report the results in Figure 4.3. We observe the following. First, we verify that increasing the timeout does indeed return higher quality solutions. The median improvement in performance goes up from 10.7% to 17.4% (`Lazy - Warm`) and 20.2% to 20.8% (`MTZ - Warm`) on low budgets, 3.5% to 40.6% (`Lazy - Warm`) and 40.3% to 43.9% (`MTZ - Warm`) on moderate budgets, and 0% to 55% (`Lazy - Warm`) and 53.4% to 58.2% (`MTZ - Warm`) on large budgets. Second, `MTZ - Warm` obtains better solutions than `Lazy - Warm` suggesting that on large instances, the MTZ formulation is preferred. The interesting fact here is that most of the performance increase is accomplished within the first 60 seconds for `MTZ - Warm`. Thus, the MIP would be useful in situations where the robot can afford to spend additional computational resources to obtain higher quality solutions (anytime property of MIPs).

4.5 Summary

In this chapter, we proposed the first exact formulation for IPP in GPs as a MIP. Our approach exploited the optimality of the posterior mean to give the resulting MIP. We showed that the approach is capable of optimally solving IPP in a variety of settings.

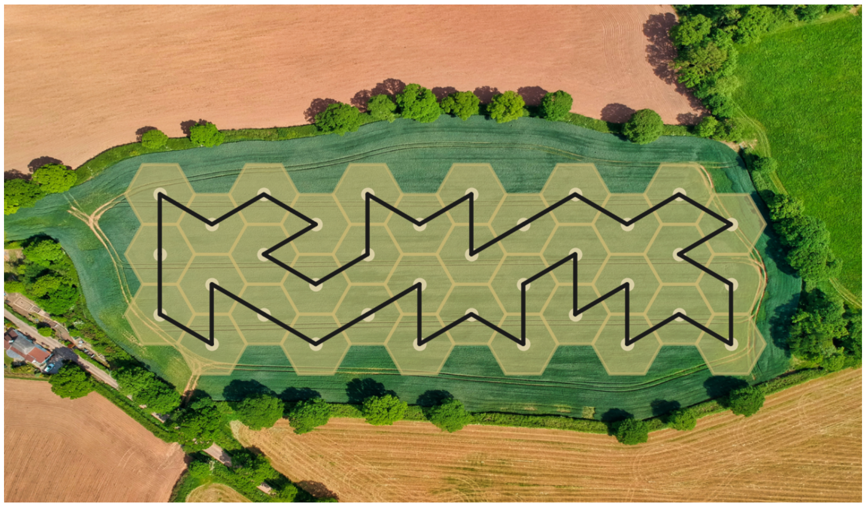
Chapter 5

Approximation Algorithms for Robot Tours in Gaussian Processes

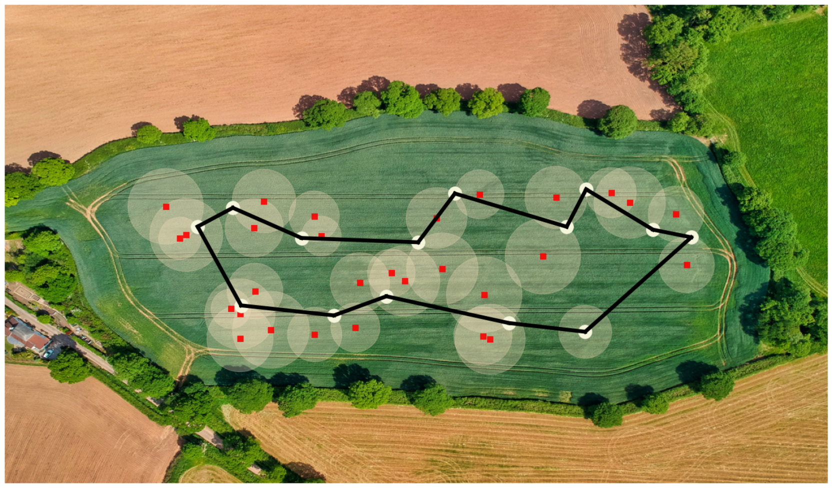
5.1 Introduction

We are often faced with the challenge of minimizing the amount of data collected in order to build models with good predictive uncertainty. This is especially important in *informative path planning* where robots must collect data in hazardous conditions where operation is expensive or risky. For example, mapping the spatial distribution of radioactivity in post-disaster environments by unmanned aerial vehicles is critical to designing measures to protect humans from radiation exposure [26]. In most cases, we can harness the structure of the phenomena to produce reliable maps with little data. This raises the question: how do we *minimize resource usage* while *reliably* mapping these environmental phenomena? In this chapter, we design approximation algorithms which produce a set of data sample locations and tours that robots can follow to ensure the resulting model predictions have low uncertainty.

We focus on the setting where the phenomena is modeled as a isotropic Gaussian Process (GP) which allows the predictive uncertainty to be measured by the posterior variance. Specifically, we consider two problems: SAMPLE PLACEMENT and SHORTEST TOUR. Both problems deal with the same constraint: ensure the posterior variance at each location in a given *test set* is within a given threshold. In SAMPLE PLACEMENT, we seek to find a candidate location set of minimum size. In a similar vein, SHORTEST TOUR aims to find the tour of minimum length whose vertex set satisfies this constraint.



(a) Convex Environments



(b) Finite Test Sets

Figure 5.1: A visualization of the solution approaches presented in this chapter for Sample Placement and Shortest Tour in two settings. The vertex set (filled white disks) of the tour (black edges) guarantee low variance in the shaded regions (yellow hexagons/white disks). The radius of the hexagons and disks are selected to ensure high prediction accuracy. a) Tours based on hexagonal covers in convex environments. b) Tours based on minimum set covers for finite test sets.

Since the general version of Sample Placement and Shortest Tour are NP-hard, we identify two cases relevant to robotics which are amenable to efficient approximation algorithms: convex environments with constant threshold variances and finite test sets with arbitrary threshold variances. Convex environments with constant thresholds are the simplest setting where one desires accurate predictions *everywhere* in the environment. However, in certain situations, we desire low predictive uncertainty only in a select few locations. For example, in precision agriculture, these locations could be regions suspected of nutrient deficiencies or pest outbreaks and the objective is to predict the quantities with high confidence.

Contributions We give four approximation algorithms as well as a heuristic for Sample Placement and Shortest Tour in two settings: convex environments and finite test sets.

- We develop HexCover and HexCoverTour for convex environments based on hexagonal covers. These improve previously known results, both in terms of the approximation factor as well as simulation results.
- We give the first approximation algorithms IntersectCover and TSPNTour for finite test sets.
- While TSPNTour is primarily a theoretical result, we give a simple heuristic IntersectTour which yields good solutions in practice.

The defining features of our algorithms are efficiency, ease of implementation, worst-case performance guarantees and use minimal assumptions when compared to prior work. Further, we also disprove a claim in the literature [122] on a necessary condition for Sample Placement in convex environments.

Related Work The sample placement and shortest tour problem for GPs with isotropic kernels have been considered previously [122, 121, 124]. The case of convex environments with constant thresholds was tackled in [122]. The authors propose a two-phase strategy for sample placement: pack the environment with large disks and then cover them with smaller disks of specific radii. Unfortunately, this two-step approach uses an excessive number of samples. In contrast, our algorithm uses fewer samples resulting from a simple covering algorithm and also improves the worst-case performance guarantees. Further, the authors claim that a packing of the larger disks provides a lower bound for the sample placement problem. However, we disprove this claim by finding a counterexample that invalidates the lower bound in Section 5.3.2.

The work in [124] solve a relaxed version of the shortest tour problem with finite test sets. The problem is, for a given set of disks, to obtain at least one measurement in every disk, termed SamplingTSPN, closely related to the traveling salesman problem with neighbourhoods [41]. The approach computes a sample set through dense grid sampling around an approximate solution to a constructed set cover problem. Further, the resulting approximation guarantee holds *only* for the relaxed problem and *not* sample placement or shortest tour. In this chapter, we provide a covering algorithm which follows in a similar spirit but has a stronger theoretical guarantee and also uses less samples in practice.

Minimizing robot resource usage to satisfy chance constraints for GP regression have also been tackled [121]. Here, the constraint is ensure the posterior distribution is concentrated around its mean with high probability. The authors in [121] give an approximation algorithm that is similar to the algorithm proposed in [122]. However, in Section 4.2, we show that this chance constraint is equivalent to the posterior variance constraint considered in this chapter. As a result, our proposed algorithms are also applicable for the chance constrained problem. Further, our algorithms also use minimal assumptions in comparison to [121]. Specifically, we do not require Lipschitz continuity on the underlying unknown function nor do we require an a priori upper bound on the solution to the sample placement problem.

5.2 Problem Formulation

We consider a metric space $(\mathcal{X}, \rho)$, where $\mathcal{X} \subset \mathbb{R}^2$ is a convex set representing the environment and $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a metric on $\mathcal{X}$. The phenomena to be estimated is modeled as an *unknown* function $f : \mathcal{X} \to \mathbb{R}$. We work in the Bayesian setting where f is a sample from a Gaussian Process with a zero mean function and an *isotropic finite range* covariance kernel $k : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with finite range $r_{\max} > 0$.

A robot collects measurements of the function f at a subset of locations in the environment $X = \{x_1, \dots, x_n\} \subset \mathcal{X}$. For each $x_i \in X$, the robot receives a measurement of the form

$$y_i = f(x_i) + \eta_i, \tag{5.1}$$

where $\eta_i \sim \mathcal{N}(0, \sigma^2)$ are zero-mean independently and identically distributed random variables. The goal is to control the posterior variance at a set of environment test locations. For any point $p \in \mathcal{X}$, define $g_p : 2^{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ to be the posterior variance at the test point p

given a subset of measurement locations in the environment $\mathcal{X}$. This is given by evaluating the posterior covariance function (2.13) at the test point p i.e.,

$$\begin{aligned} g_p(X) &:= \bar{k}(p,p) \\ &= k(p,p) - k_{pX}(k_{XX} + \sigma^2 I_n)^{-1} k_{Xp}. \end{aligned} \tag{5.2}$$

We consider two problems closely related to each other. The input to both is the tuple $(\mathcal{X}, \rho, P, k, w, r_{\max})$ where $(\mathcal{X}, \rho)$ is a metric space, $P \subseteq \mathcal{X}$ is the set of test locations, k is the kernel of the underlying zero mean GP, and $w : P \to \mathbb{R}_{\geq 0}$ returns the threshold variance for any test point.

Problem 6 (Sample Placement)**.** Find the minimum cardinality measurement subset that satisfies the posterior variance constraint i.e.,

$$\begin{aligned} \underset{S \subset \mathcal{X}}{\text{minimize}} \quad & |S| \\ \text{subject to} \quad & g_p(S) \leq w(p),\ p \in P. \end{aligned} \tag{5.3}$$

Problem 7 (Shortest Tour)**.** Let $\mathcal{T}$ be the set of all tours in the environment $\mathcal{X}$. Then, we seek the minimum length tour whose vertex set satisfies the posterior variance constraint i.e.,

$$\begin{aligned} \underset{T \subset \mathcal{T}}{\text{minimize}} \quad & \text{length}(T) \\ \text{subject to} \quad & g_p(T) \leq w(p),\ p \in P. \end{aligned} \tag{5.4}$$

•

Generality of Formulation Chance constrained regression (CCR) in GPs can be reformulated into the posterior variance constraint considered in the problems above. As a result, the algorithms proposed in this chapter can be used for CCR as well. In CCR with GPs [121, 124], the goal is to ensure the posterior distribution is concentrated around its mean with high probability. More formally, given $\epsilon, \delta > 0$ and a test set $P \subset \mathcal{X}$, for any sample locations $X = \{x_1, \ldots, x_n\} \in \mathcal{X}^n$ and the associated measurements $Y = \{y_1, \ldots, y_n\} \in \mathbb{R}^n$ given by (5.1), the constraint at a given test point $p \in P$ is

$$\mathbb{P}\Big[\big|f(p) - \mathbb{E}[f(p)|Y]\big| \leq \epsilon \Big| Y\Big] \geq \delta. \tag{5.5}$$

Using Theorem 3, we have that $f(p)|Y \sim \mathcal{N}(\bar{m}(p), \bar{k}(p,p))$ i.e., $f(p)|Y$ is a Gaussian random variable with mean $\mathbb{E}[f(p)|Y] = \bar{m}(p)$ and variance $\mathrm{Var}(f(p)|Y) = \bar{k}(p,p)$. To reformulate the constraint in terms of the posterior variance, we require the following lemma.

Lemma 6. Let $X \sim \mathcal{N}(\mu, \sigma^2)$. Then, for any $\epsilon, \delta > 0$,

$$\mathbb{P}\Big[|X - \mu| \leq \epsilon\Big] \geq \delta \iff \mathrm{Var}(X) \leq \left(\frac{\epsilon}{\phi^{-1}(\frac{\delta}{2})}\right)^2, \tag{5.6}$$

where ϕ is the CDF for the standard normal.

Now, applying Lemma 6 with $X = f(p)|Y, \mu = \bar{m}(p)$, and $\sigma^2 = \bar{k}(p,p)$ allows the reformulation of (5.5) as

$$\bar{k}(p,p) \leq \left(\frac{\epsilon}{\phi^{-1}(\frac{\delta}{2})}\right)^2. \tag{5.7}$$

Using the definition of $g_p(X)$ in (5.2) and setting $w(p) = \Big(\epsilon/\phi^{-1}(\delta/2)\Big)^2$, for all $p \in P$, we recover the posterior variance constraint considered in this chapter.

5.3 Solution Approach

The general version of Sample Placement and Shortest Tour are NP-hard. As a result, we identify two cases which are amenable to approximation algorithms: convex environments with constant threshold variances (Section 5.3.2) and finite test sets with arbitrary thresholds (Section 5.3.3). Both algorithms rely on a sufficient condition on a distance outlined in Section 5.3.1.

5.3.1 Sufficient Conditions

The central idea in both algorithms is to approximately compute minimum size covers of the points in the test set $P \subseteq \mathcal{X}$ and subsequently plan tours on these sampled points. Specifically, each test point $p \in P$ will have a sample $x \in \mathcal{X}$ in the environment within a specified distance $r^p_{\min}$. With an appropriate selection of $r^p_{\min}$, this will ensure the posterior variance constraint is satisfied thereby yielding feasible solutions to Sample Placement and Shortest Tour.

The distance $r^p_{\min}$ will depend on the choice of the kernel function. For most kernel functions, one can find a closed-form expression for the distance. For example, the following result gives the expression for the distance when using the squared exponential (SE) kernel $k(d) = \sigma_0^2 e^{-\frac{d^2}{2L^2}}$.

Lemma 7 (Sufficient Condition for SE Kernel)**.** Let $L > 0, \sigma_0 > 0$ be the parameters of the squared exponential kernel. Next, for each test point $p \in P$, define the distance

$$r^p_{\text{min}} := L\sqrt{-\log\left((\sigma_0^2 - w(p))\frac{\sigma_0^2 + \sigma^2}{\sigma_0^4}\right)}. \tag{5.8}$$

Then, a set $S \subset \mathcal{X}$ is feasible for Sample Placement if for each $p \in P$, there exists $x \in S$ such that $\rho(x, p) \leq r^p_{\text{min}}$.

Proof. We will show that for each test point $p \in P$, the posterior variance $g_p(S) \leq w(p)$. We are given that there exists a point $x \in S$ such that $\rho(x, p) \leq r^p_{\text{min}}$. In addition, since $g_p(S)$ is a monotonically decreasing set function [77], it suffices to show $g_p(\{x\}) \leq w(p)$. Then,

$$\begin{aligned}\rho(x,p) \leq r^p_{\text{min}} &\implies e^{-\frac{\rho(x,p)^2}{L^2}} \geq (\sigma_0^2 - w(p))\frac{\sigma_0^2 + \sigma^2}{\sigma_0^4} \\ &\implies g_p(\{x\}) \leq w(p).\end{aligned}$$

$\square$

One can use similar ideas to find the required distances for other isotropic kernels [133]. If a certain kernel prevents one from computing a closed-form expression for r^p_{min}, we can execute binary search[1] on the interval $[0, r_{\text{max}}]$. Specifically, place a point $x \in \mathcal{X}$ at a distance $r = r_{\text{max}}/2$ from a test point p and check if $g_p(\{x\}) \leq \Delta(p)$. If satisfied, then $r^p_{\text{min}} \in [r, r_{\text{max}}]$ else $r^p_{\text{min}} \in [0, r]$. Now that r^p_{min} has been fully specified for each test point $p \in P$, we proceed to discuss how to generate these covers in the continuous (convex environments) and discrete setting (finite test sets).

5.3.2 Convex Environments, Constant Thresholds

The setting considered in this section is $P = \mathcal{X}$ with the threshold variance for each point being constant i.e., $w(p) = \Delta$[2]. The three contributions in this section are: a) disproving a claimed lower bound in [122] on the optimal solution for Sample Placement, b) HexCover for Sample Placement, and c) HexCoverTour for Shortest Tour. We will denote the optimal solutions to Sample Placement and Shortest Tour by S^* and T^* respectively.

[1]This works only if the kernel is monotonically decreasing as a function of the distance which is the case for most kernels.

[2]Our results also work when P is a convex subset of the environment $\mathcal{X}$.

Disproving a Necessary Condition The approach for SAMPLE PLACEMENT proposed in [122] is called DISKCOVER and uses two steps. First, it covers the environment with large circles of a certain radius $r_{\max}$. Second, it then covers these circles with smaller circles to obtain a feasible solution which yields an approximation algorithm. The analysis does not use $r_{\max}$, the finite range assumption on the kernel of the GP. Instead, it involves a claim on a necessary condition for the problem. We provide a counterexample to disprove this claim.

Without the finite range assumption, it would be useful to obtain a distance R such that if a test location does not have a sample within R, then the posterior variance constraint cannot be satisfied. Then, a packing of circles of radius R is a lower bound on the optimal value to SAMPLE PLACEMENT . This is the idea behind the necessary condition [122]. We state the lemma below.

Lemma 8 (Lemma 1, [122])**.** For any test location x, if the nearest measurement location is at a distance R away, and

$$R > L\sqrt{-\log\left(1 - \frac{\Delta}{\sigma_0^2}\right)}, \tag{5.9}$$

then it is not possible to bring down the posterior variance below Δ at x.

The proof constructs a bound for the posterior variance by placing all samples at the nearest location to the test location x. This is shown in the left plot in Figure 5.2. While this seems intuitive, this does not give a valid lower bound on the variance. We construct a measurement set, as shown in the right plot in Figure 5.2, that obtains a lower variance.

Example 1. Consider the following environment setup: $\sigma = 1, \sigma_0 = 1, L = 1, x = (0,0), \Delta = 0.5$. Then, the RHS of (5.9) is 0.83255461. Let $R = 0.93255461$ and consider the following measurement set:

$$S := \{(R,0), (-R,0), (0,R), (0,-R)\}. \tag{5.10}$$

The posterior variance at x using measurement set S is

$$f_x(S) = 0.443771 < \Delta, \tag{5.11}$$

which shows the contradiction. •

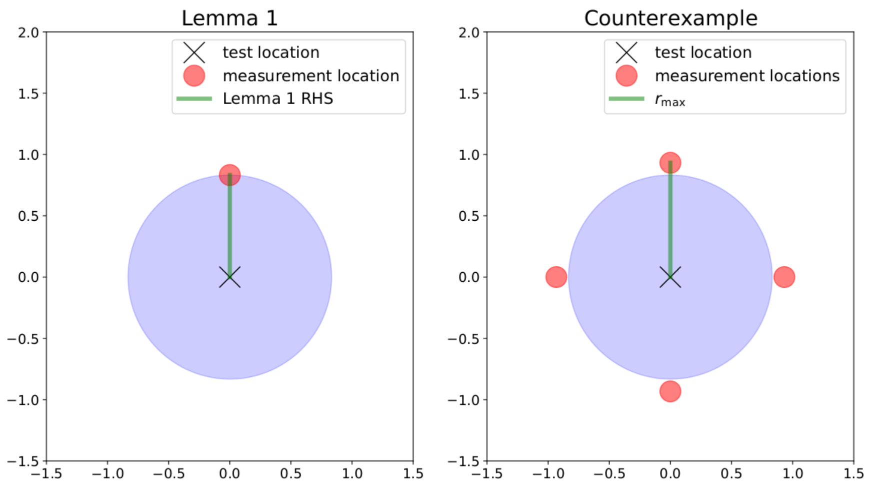


Figure 5.2: Visual depiction of the counterexample. Left: Lemma 1 in [122], constructs a lower bound by placing samples at the red circle. Right: The set of four measurement locations obtains a lower posterior variance.

The counterexample uses four points to invalidate the lower bound on R. However, by placing a large number of samples outside R, the variance can be much lower, thereby underestimating the lower bound on R by a large margin. By using the finite range assumption on the kernel of the GP, which is a common practical assumption in geostatistics [133], the approximation ratios in [122] still hold (though Lemmas 1 and 3 in [122] do not hold). Since the proposed algorithm [122] covers the environments with circles of radius R (which can be large), and then covers it with many smaller circles, it uses an excessive number of samples. This motivates us to develop an alternative approach that uses less samples.

HexCover Our approach to solving Sample Placement in this setting is to produce a $r_{\min}$-covering of the environment $\mathcal{X}$ with circles of a specific radius. Since we are dealing with a constant threshold Δ, we drop the dependence of $r^p_{\min}$ on the test point p and use $r_{\min}$ for the remainder of this section. We have already seen that the selection of $r_{\min}$ (Section 5.3.1) guarantees feasibility.

The last step is to generate the $r_{\min}$-covering. Since the objective is to minimize the number of samples, one idea is to compute the minimum $r_{\min}$-covering. Unfortunately, this is an NP-hard problem [51]. Instead, we compute a cover from a hexagonal tiling, which is the densest way to pack circles in the Euclidean plane [50, 21]. The steps are described in Algorithm 2. The main step is HexagonalTiling (Line 2), which returns the set of centers in the hexagonal tiling of edge length $r_{\min}$. The centers are arranged in staggered columns where the vertical distance between two samples is $\sqrt{3}r_{\min}$ and the horizontal spacing between the columns is $1.5r_{\min}$. An example of a hexagonal tiling is shown in Figure 5.1a. By construction, the circles centered at these points will circumscribe the hexagon, thereby covering the environment.

Algorithm 2: HexCover

```
    Input: Environment $\mathcal{X} \subset \mathbb{R}^2$, Variance Threshold: $\Delta$
    Output: Measurement Set: $S \subset \mathcal{X}$
1   Compute $r_{\min}$ according to Section 5.3.1.
2   $S_{\mathrm{HC}}$ = HexagonalTiling($\mathcal{X}, r_{\min}$)
3   return $S_{HC}$
```

Assumption 1 (Boundary Conditions)**.** For some environments, there exist regions close to the boundary that are not covered by the hexagonal tiling. This is solved by taking a few more measurements to ensure feasibility. However, for the purposes of algorithm analysis, we will assume that the hexagonal tiling (Line 2) covers the environment completely.

Analysis: To analyse the performance of this approach, the first step is to compute an upper bound on the solution $|S_{\mathrm{HC}}|$ returned by HexCover. To do this, we require the definition of a Minkowksi sum and a lemma bounding the maximum area of the Minkowski sum of a convex set and a unit circle. Our proof of this result is based on the lecture notes [136] and is included here for completeness.

Let $B := \{(x, y) : x^2 + y^2 \leq 1\}$ be the unit circle and for any convex set Θ and $c > 0$, define $c\Theta := \{cx : x \in \Theta\}$.

Definition 5 (Minkowski Sum)**.** For any two sets A, B, the Minkowski sum is $A + B := \{x + y : x \in A, y \in B\}$.

Lemma 9. *Given a convex set $\Theta \subset \mathbb{R}^2$, the unit circle B, and $r > 0$ with $rB \subset \Theta$, then*

$$\text{area}\left(\Theta + \frac{r}{2}B\right) \leq \frac{9}{4}\text{area}\left(\Theta\right).$$

Proof. Take any $a \in \Theta + \frac{r}{2}B$. Then, it can be expressed as $a = x + y$, where $x \in \Theta, y \in \frac{r}{2}B$. Since $rB \subset \Theta$, then $\frac{r}{2}B \subset \frac{1}{2}\Theta$. Thus, $a \in \Theta + \frac{1}{2}\Theta$. Now, a can be expressed as $a = s + \frac{t}{2}$, where $s, t \in \Theta$. Then, $\frac{2}{3}a = \frac{2}{3}s + \frac{t}{3}$. Since Θ is convex, we have $\frac{2}{3}a \in \Theta$ which implies $a \in \frac{3}{2}\Theta$. Thus, $\Theta + \frac{r}{2}B \subset \frac{3}{2}\Theta$ from which the final result follows. □

Based on Lemma 9 we now establish an upper bound on the number of measurements $|S_{\mathrm{HC}}|$.

Lemma 10 (Upper Bound)**.** Let $\delta > 0$ be arbitrarily small and let $(\sqrt{3}-\delta)r_{\min}B \subset \mathcal{X}$, where B is the unit circle. Then, for every $\epsilon_\delta > 0$,

$$|S_{\mathrm{HC}}| \leq (3+\epsilon_\delta)\frac{\mathrm{area}(\mathcal{X})}{\pi r_{\min}^2}.$$

Proof. Denote the solution $S_{\mathrm{HC}} := \{x_1, \ldots, x_k\}$. Then, under Assumption 1, S_{HC} is a $(\sqrt{3}-\delta)r_{\min}$-packing for the environment $\mathcal{X}$. This is because the minimum distance between any two points in the hexagonal cover is $\sqrt{3}r_{\min}$ i.e., $\min_{i\neq j}\rho(x_i, x_j) = \sqrt{3}r_{\min} > (\sqrt{3}-\delta)r_{\min}$, which ensures it is a $(\sqrt{3}-\delta)r_{\min}$-packing. Since the circles in the packing are disjoint, we have $\bigcup_i^k B(x_i, \frac{(\sqrt{3}-\delta)}{2}r_{\min}) \subset \mathcal{X} + \frac{(\sqrt{3}-\delta)}{2}r_{\min}B$. Then, letting $c = \sqrt{3}-\delta$ and taking the area on both sides gives

$$\begin{aligned}
\mathrm{area}\left(\bigcup_i^{|S_{\mathrm{HC}}|} B(x_i, cr_{\min}/2)\right) &= |S_{\mathrm{HC}}|\frac{\pi c^2 r_{\min}^2}{4} \\
&\leq \mathrm{area}\left(\mathcal{X} + \frac{cr_{\min}}{2}B\right) \leq \frac{9}{4}\mathrm{area}(\mathcal{X}) \\
\implies |S_{\mathrm{HC}}| &\leq \left(\frac{3}{c}\right)^2 \frac{\mathrm{area}(\mathcal{X})}{\pi r_{\min}^2} \\
&= (3+\epsilon_\delta)\frac{\mathrm{area}(\mathcal{X})}{\pi r_{\min}^2},
\end{aligned} \tag{5.12}$$

where the second inequality follows from Lemma 9 and

$$\epsilon_\delta = \left(\frac{3}{\sqrt{3}-\delta}\right)^2 - 3$$

is arbitrarily small. □

To obtain an approximation factor for Algorithm 2, we characterize an optimal solution S^*. First, we establish a property on any feasible solution which we will then use to get a lower bound on $|S^*|$.

Lemma 11 (Feasible Sample Placement)**.** Any feasible solution S for Sample Placement is a $r_{\max}$-covering of $\mathcal{X}$.

Proof. Suppose not. Then, there exists $x \in \mathcal{X}$ such that for all $y \in S$, $\rho(x, y) > r_{\max}$. Using the finite range $r_{\max}$, the posterior variance at x using S is $f_x(S) = \sigma_0^2 > \Delta$, contradicting feasibility of S for Sample Placement. □

Lemma 12 (Lower Bound)**.** A lower bound on the optimal value $|S^*|$ for Sample Placement is

$$|S^*| \geq \frac{\text{area}(\mathcal{X})}{\pi r_{\max}^2}.$$

Proof. Using Lemma 11, S^* is a $r_{\max}$-covering of the environment $\mathcal{X}$ i.e., $\bigcup_{x \in S^*} B(x, r_{\max}) \supset \mathcal{X}$. Taking area on both sides gives

$$\begin{aligned}\text{area}(\mathcal{X}) &\leq \text{area}\left(\bigcup_{x \in S^*} B(x, r_{\max})\right) \\ &\leq \sum_{x \in S^*} \text{area}\left(B(x, r_{\max})\right) = |S^*|\pi r_{\max}^2,\end{aligned}$$

where the second inequality follows from the fact that area is a sub-additive function. □

Theorem 7. For any $\Delta > 0$, if $w(p) = \Delta$ for each $p \in \mathcal{X}$ and $\mathcal{X}$ is convex, then for every $\epsilon > 0$, HexCover is an α-approximation algorithm for Sample Placement where

$$\alpha := 3\frac{r_{\max}^2}{r_{\min}^2} + \epsilon.$$

Proof. Using Lemmas 10 and 12, for some arbitrarily small $\epsilon_\delta > 0$, $|S_{\text{HC}}| \leq (3+\epsilon_\delta)\frac{\text{area}(\mathcal{X})}{\pi r_{\min}^2} \leq (3 + \epsilon_\delta)\frac{r_{\max}^2}{r_{\min}^2}|S^*| = (3\frac{r_{\max}^2}{r_{\min}^2} + \epsilon)|S^*|$, where $\epsilon = \epsilon_\delta \frac{r_{\max}^2}{r_{\min}^2}$ is arbitrarily small. □

Theorem 7 improves the previous approximation ratio of $18\frac{r_{\max}^2}{r_{\min}^2}$ [122]. Our approximation factor is independent of the environment $\mathcal{X}$ but does depend on the variance threshold Δ through $r_{\min}$. As we seek more accurate predictions (decreasing Δ), the radius of accurate estimation $r_{\min}$ reduces, and the number of measurements required will increase.

HexCoverTour Our approach to solving the shortest tour problem builds on HexCover. This is because the constraints are the same in both problems. Since we have already identified a feasible vertex set via HexCover, we will use an approximation algorithm for the metric TSP, Christofides' algorithm, which finds a tour of length no more than 3/2 times the optimal [76]. The steps are outlined in Algorithm 3.

Algorithm 3: HexCoverTour

```
    Input: Environment $\mathcal{X} \subset \mathbb{R}^2$, Variance Threshold: $\Delta$
    Output: Tour $T$ with $T \subset \mathcal{X}$.
1   $S_{\text{HC}} = \text{HexCover}(\mathcal{X}, \Delta)$
2   $T_{\text{HC}} = \text{ChrisotofidesTour}(S_{\text{HC}})$
3   return $T_{HC}$
```

We will now bound the worst-case performance of this algorithm. We begin by characterizing an upper bound on the length of the tour T_{HC} produced by HexCoverTour.

Lemma 13 (Upper Bound)**.** Let $\mathcal{X}$ be a convex set. For every $\epsilon > 0$,

$$\text{len}(T_{\text{HC}}) \leq 15.6\frac{\text{area}(\mathcal{X})}{\pi r_{\min}} + \epsilon.$$

Proof. Denote the vertex set of the tour by $V_{\text{HC}} := \{x_1, \ldots, x_k\}$ and let T_k denote the optimal TSP tour on the k vertices of the tour T_{HC}. Note that in the hexagonal covering, the minimum distance between two points is $\sqrt{3}r_{\min}$. Now, we can bound T_k by constructing a sub-optimal tour as follows. Take any two points $x_i, x_j \in V_{\text{HC}}$ such that $\rho(x_i, x_j) \leq \sqrt{3}r_{\min}$ and find the shortest tour in $V_{\text{HC}} \setminus x_i$. Then, we can create a tour by adding the edge to connect x_j to x_i and back. This gives an upper bound:

$$\text{len}(T_k) \leq \text{len}(T_{k-1}) + 2\sqrt{3}r_{\min}.$$

Using the fact that $\text{len}(T_1) = 0$ and Lemma 10, for any ϵ_δ, we get

$$\text{len}(T_k) \leq 2\sqrt{3}r_{\min}|V_{\text{HC}}| \leq 2\sqrt{3}(3 + \epsilon_\delta)\frac{\text{area}(\mathcal{X})}{\pi r_{\min}}.$$

Since we are using Christofides' algorithm, we have

$$\begin{aligned}\text{len}(T_{\text{HC}}) \leq 1.5\text{len}(T_k) &\leq 3\sqrt{3}(3 + \epsilon_\delta)\frac{\text{area}(\mathcal{X})}{\pi r_{\min}} \\ &= 15.6\frac{\text{area}(\mathcal{X})}{\pi r_{\min}} + \epsilon,\end{aligned}$$

where $\epsilon = 3\sqrt{3}\frac{\text{area}(\mathcal{X})}{\pi r_{\min}}\epsilon_\delta$ is arbitrary. □

In the following lemma, we compute a lower bound on the length of the optimal tour T^*. First, we need a property of a maximal packing of a $r_{\max}$-covering.

Lemma 14. Let $S := \{x_1, \dots, x_n\}$ be a $r_{\max}$-covering of the environment $\mathcal{X}$. Then, any maximal $2r_{\max}$-packing $P \subset S$ of S is a $3r_{\max}$-covering of $\mathcal{X}$.

Proof. Using the triangle inequality and the fact that S is a $r_{\max}$-covering of $\mathcal{X}$, it suffices to show that for any $x_i \in S$, there exists a point in the packing $y \in P$ such that $\rho(x_i, y) \leq 2r_{\max}$. Suppose not. Then, there exists $x_i \in S$ such that for all $y \in P$, $\rho(x_i, y) > 2r_{\max}$. But, we could add x_i to P and increase its size, contradicting the fact it is maximal. □

Lemma 15 (Lower Bound)**.** A lower bound on the length of the optimal tour is

$$\operatorname{len}(T^*) \geq \frac{2}{9} \frac{\operatorname{area}(\mathcal{X})}{\pi r_{\max}}.$$

Proof. We compute a maximal disjoint set $T' \subset T^*$ from the set of circles of radius $r_{\max}$ centered at points in the optimal tour T^*. By the triangle inequality, the optimal TSP tour through T' gives us a bound on the length of the optimal tour:

$$\operatorname{len}(T^*) \geq \operatorname{len}(T'). \tag{5.13}$$

Since T' is a $2r_{\max}$-packing, the minimum distance between any two vertices in T' is $2r_{\max}$. Thus,

$$\operatorname{len}(T') \geq 2r_{\max}|T'|. \tag{5.14}$$

Since T^* is a $r_{\max}$-covering of $\mathcal{X}$ (Lemma 11), T' is a $3r_{\max}$-covering (Lemma 14). Then, $\bigcup_{x \in T'} B(x, 3r_{\max}) \supset \mathcal{X}$ which implies

$$\begin{aligned} \operatorname{area}\left(\bigcup_{x \in T'} B(x, 3r_{\max})\right) &\geq \operatorname{area}(\mathcal{X}) \\ \implies \sum_{x \in T'} \operatorname{area}\left(B(x, 3r_{\max})\right) &\geq \operatorname{area}(\mathcal{X}) \\ \implies |T'| &\geq \frac{\operatorname{area}(\mathcal{X})}{9\pi r_{\max}^2}. \end{aligned} \tag{5.15}$$

Combining Equations (5.13), (5.14), and (5.15) gives us the result. □

The guarantee is a consequence of Lemmas 13 and 15.

Theorem 8. Given $\Delta > 0$, if $w(p) = \Delta$ for each $p \in \mathcal{X}$ and $\mathcal{X}$ is convex, then HexCoverTour is an α-approximation algorithm for Shortest Tour where for an arbitrarily small $\epsilon > 0$,

$$\alpha := 70.2\frac{r_{\max}}{r_{\min}} + \epsilon.$$

This improves the previous approximation ratio of $9.33 + O(\frac{r_{\max}^2}{r_{\min}^2})$ [122]. Note that in practice, one will either use an optimal TSP solver (if the instance is small enough) or heuristics such as the Lin-Kernighan Heuristic [76, 66] which are known to be near-optimal. This would roughly improve the performance guarantee by 1.5 times leading to $46.8\frac{r_{\max}}{r_{\min}} + \epsilon$ but no longer have guarantees on running time.

5.3.3 Finite Test Sets, Arbitrary Thresholds

In this section, we study the case of a finite test set. We denote it by $P := \{p_1, \ldots, p_n\}$ of size n with variance thresholds $w(p_1), \ldots, w(p_n)$. We develop approximation algorithms IntersectCover and TSPNTour for Sample Placement and Shortest Tour respectively. TSPNTour uses the algorithm in [42] which is an involved procedure and has a large time complexity. In light of this, we develop a heuristic IntersectTour that is fast and produces good results in simulation but does not have an approximation guarantee.

Recall that we computed $r_{\min}^p$ for each $p \in P$ in Section 5.3.1. We denote the maximum and minimum radii by

$$\begin{aligned} r_{\min}^{[n]} &:= \max\{r_{\min}^p : p \in P\}, \\ r_{\min}^{[1]} &:= \min\{r_{\min}^p : p \in P\}. \end{aligned} \tag{5.16}$$

IntersectCover First, we study how we can place samples to address finite test sets. The high level idea is to ensure each test point has a sample sufficiently close to it. We will accomplish this by converting the problem into an instance of `SetCover-D`.

In Figure 5.3, we visualize an instance of `SetCover-D`$(\mathcal{B})$ for some set of disks $\mathcal{B}$. Each disk denotes a test point $p_i \in P$ (in orange) and an associated radius r_i. We denote the regions of the arrangement by R_i. For example, R_2 is the intersection of disks centered at p_1, p_3, p_4 and R_1 is the intersection of the disks at p_1 and p_4 but without region R_2 i.e., $R_2 = B(p_1, r_{\min}^{p_1}) \cap B(p_3, r_{\min}^{p_3}) \cap B(p_4, r_{\min}^{p_4})$ and $R_1 = B(p_1, r_{\min}^{p_1}) \cap B(p_4, r_{\min}^{p_4}) \setminus R_2$. We want

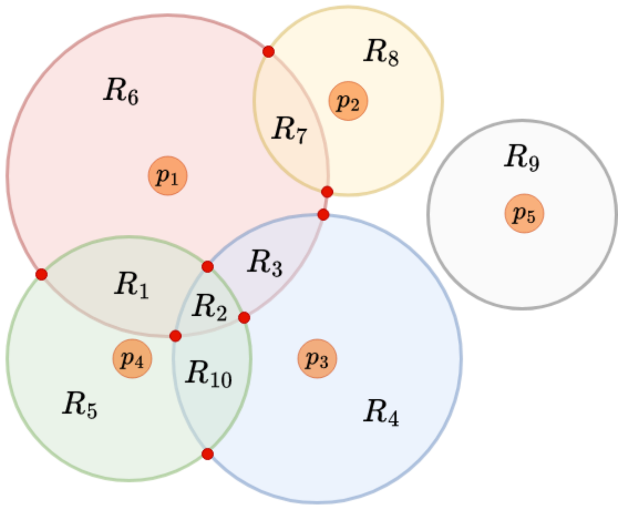


Figure 5.3: An example of a `SetCover-D` instance. The objective is to place the minimum number of samples such that each disk contains at least one sample. The size of the optimal solution for this instance is 3.

to find the minimum number of samples that cover each test point i.e., a sample in each disk. It is easy to see the optimal solution has size 3 (for example, sample in R_2, R_7, R_9).

The goal is to solve `SetCover-D` on the set of disks centered at P with radii $r_{\text{min}}^{p_i}$. However, finding a minimum size cover for disks on a subset of $\mathbb{R}^2$ is not obvious. Instead of solving `SetCover-D` directly, we will reduce the search space to a finite set of candidate locations without losing optimality of the resulting solution i.e., we will solve `SetCover` on appropriately constructed sets. Recall that a set cover instance is described by a tuple $(\mathcal{U}, \mathcal{D})$. The universe of elements is set to the test set $\mathcal{U} = P$. We outline the steps to generate the collection of subsets $\mathcal{D}$ in Algorithm 4. The idea is to only consider points that are i) the intersection of the boundaries of two disk, or ii) the center of isolated disks, i.e., disks that do not intersect other disks (Lines 2-8). Then, for each resulting point, we compute the subset of test points it covers (Lines 9-16). Specifically, for each disk, we find the intersecting points with all other disks (Line 5). The number of intersecting points can be zero (no intersection), one (circles are tangent to each other), or two (intersect at distinct points). For example, the intersections points in Figure 5.3 are shown in red. Then, for each collected point and for each test point, we check whether it is covered (Line 14). If so, we add it to the subset corresponding to that sample. Finally, we return the collection of subsets as well as the corresponding samples (Line 17).

By solving `SetCover`$(P, \mathcal{D})$, we obtain a solution to `SetCover-D`. The approach is out-

Algorithm 4: GenSubset

```
Input: Instance of Sample Placement $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$
Output: Subset collection of $P$: $\mathcal{D} = \{D_1, \ldots, D_m\}$, candidate sample locations:
        samples
1  samples = empty
2  for $i = 1, \ldots, n$ do
3      pts = empty
4      for $j = i + 1, \ldots, n$ do
5          compute $r^{p_i}_{\min}, r^{p_j}_{\min}$ according to Section 5.3.1
6          add intersecting points of disks $B(p_i, r^{p_i}_{\min})$, $B(p_j, r^{p_j}_{\min})$ to pts
7      if pts is empty then
8          pts = $p_i$
9      add pts to samples
10 $\mathcal{D}$ = empty
11 $m$ = |samples|
12 for $i = 1, \ldots, m$ do
13     $D_i$ = empty
14     for $j = 1, \ldots, n$ do
15         if $\rho(samples[i], p_j) \leq r^{p_j}_{\min}$ then
16             add $p_j$ to $D_i$
17     add $D_i$ to $\mathcal{D}$
18 return $\mathcal{D}$, samples
```

lined in Algorithm 5. First, we compute the candidate samples and associated subsets (Line 1). Then, we approximately solve `SetCover`$(P, \mathcal{D})$ (Line 2) using an approximation algorithm ([49] or [1]). Finally, we get the samples associated with the selected subsets (Lines 3-5) and return the solution. We now characterize the runtime. Define γ to be the maximum number of disks any one disk can intersect with in the set $\{B(p_i, r_{\min}^{p_i}) : i \in [n]\}$. Then, GenSubset runs in time $O(\gamma n^2)$ and IntersectCover runs in time $O(\gamma n^2 + g(n))$ where $g(n)$ is the time complexity of an approximation algorithm for `SetCover`. For example, the greedy algorithm [49] has a runtime of $O(nm)$ where the number of subsets $m = O(\gamma n)$. This results in a runtime of $O(\gamma n^2)$ for IntersectCover.

Algorithm 5: IntersectCover

```
  Input: Instance of Sample Placement $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$
  Output: Measurement Set $S \subset \mathcal{X}$
1 $\mathcal{D}$, samples = GenSubset($I$)
2 $I$ = ApproximateCover($P, \mathcal{D}$)
3 $S$ = samples[$I$]
4 return $S$
```

We first present the approximation guarantee of IntersectCover and give the proof later in this section.

Theorem 9. Define $\omega := \frac{r_{\max}}{r_{\min}^{[1]}}$. For a finite test set P, IntersectCover is an $O(\omega^2)$-approximation for Sample Placement.

Remark (Comparison with [124])**.** We note that the work in [124] developed algorithms to solve `SetCover-D` and `TSPN-D` and *not* Sample Placement or Shortest Tour. Given a β-approximation algorithm for `SetCover`, [124] provides a 25β-approximation algorithm for `SetCover-D`. In contrast, using Lemma 17 below, we get a β-approximation for `SetCover-D` improving the previous factor by 25.

For the analysis, we will require definitions of a few key quantities. For any problem instance $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$ where $P = \{p_1, \ldots, p_n\}$ is finite, define the following sets

$$\begin{aligned} \mathcal{B}^I_{r_{\min}} &:= \{B(p_i, r_{\min}^{p_i}) : i \in [n]\}, \\ \mathcal{B}^I_{r_{\max}} &:= \{B(p_i, r_{\max}) : i \in [n]\}, \end{aligned} \tag{5.17}$$

where the radii $r_{\min}^{p_i}$ are computed using the sufficient condition in Section 5.3.1.

Let $\mathcal{D}_I$ be the subsets generated by GenSubset(I) with $m = |\mathcal{D}_I|$. We denote the optimal solutions to a few selected problems operating on instance I below.

$$\begin{aligned} S^*_{r_{\max}}(I) &= \mathrm{OPT}\Big(\texttt{SetCover-D}(\mathcal{B}^I_{r_{\max}})\Big) \subset \mathcal{X}, \\ S^*_{r_{\min}}(I) &= \mathrm{OPT}\Big(\texttt{SetCover-D}(\mathcal{B}^I_{r_{\min}})\Big) \subset \mathcal{X}, \\ J^*(I) &= \mathrm{OPT}\Big(\texttt{SetCover}(P, \mathcal{D}_I)\Big) \subseteq [m]. \end{aligned} \tag{5.18}$$

We need to establish that for any instance I, GenSubset generates the relevant subsets $\mathcal{D}_I$ for `SetCover`. In the following result, we show that for any point x in the environment, there exists a subset in $\mathcal{D}_I$ that covers at least as many test points as x.

Lemma 16. Let $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$ be a problem instance where $P = \{p_1, \dots, p_n\}$ is finite. Further, let $\mathcal{D}_I$ be the subset generated by GenSubset(I). For any $x \in \mathcal{X}$, denote $D_x := \{p \in P : x \in B(p, r^p_{\min})\} \subseteq P$ as the subset of test points whose associated disks contain x. Then, for any $x \in \mathcal{X}$ with $|D_x| \geq 1$, there exists $y_x \in \mathcal{X}$ such that $D_{y_x} \in \mathcal{D}_I$ and $D_x \subseteq D_{y_x}$.

Proof. We will prove this by induction on the size $|D_x|$. Let $P(m)$ be the statement that for any $x \in \mathcal{X}$ with $|D_x| = m$ and $m \geq 1$, there exists $y \in \mathcal{X}$ such that $D_{y_x} \in \mathcal{D}_I$ and $D_x \subseteq D_{y_x}$.

To prove the condition $D_{y_x} \in \mathcal{D}_I$, it is sufficient to show the existence of $y_x \in \mathcal{X}$ such that either $y_x \in P$ is a test point whose associated disk boundary $\mathrm{bd}(B(y_x, r^{y_x}_{\min})$ does not intersect the boundary of any other disk in $\mathcal{B}^I_{r_{\min}}$ or that it is an intersection point between the boundaries of two disks i.e., $\exists i \neq j \in [n], y \in \mathcal{X}$ such that $y \in \mathrm{bd}(B(p_i, r^{p_i}_{\min})) \cap \mathrm{bd}(B(p_j, r^{p_j}_{\min}))$. This follows from how the subsets in $\mathcal{D}_I$ are computed in GenSubset.

Base case ($m = 1$): $|D_x| = 1$. Let $D_x = \{p_i\}$.

- If $\mathrm{bd}(B(p_i, r^{p_i}_{\min}))$ does not intersect the boundary of any other disk in $\mathcal{B}^I_{r_{\min}}$, set $y_x = p_i$. Then, $D_x = D_{y_x}$.
- If $\mathrm{bd}(B(p_i, r^{p_i}_{\min}))$ intersects with at least one other disk boundary i.e., $\exists j \in [n]$ such that $A_{ij} = \mathrm{bd}(B(p_i, r^{p_i}_{\min})) \cap \mathrm{bd}(B(p_j, r^{p_j}_{\min})) \neq \emptyset$, then set y_x to be any element of A_{ij}. Then, $D_x \subset D_{y_x}$.

We assume the statement $P(k)$ is true for some positive integer k. We will show $P(k+1)$ is true. We are given $x \in \mathcal{X}$ with $|D_x| = k+1$ i.e., x lies in the intersection of $k+1$ disks. Consider any subset $I \subset D_x \subset P$ of size $|I| = k$ and let $p' = D_x \setminus I$. Define $R_I := \cap_{p \in I} B(p, r^p_{\min})$. Note that R_I is the region of intersection of k disks and $R_{D_x} := R_I \cap B(p', r^{p'}_{\min})$ is the intersection of all $k+1$ disks. We consider two cases.

- $R_{D_x} = R_I$: Since $x \in R_I$ and $|I| = k$, using the induction hypothesis, there exists y_x such that $I \subseteq D_{y_x}$ and $D_y \in \mathcal{D}_I$. Since $R_{D_x} = R_I$, y_x also intersects $B(p', r^{p'}_{\min})$. Then, $D_{y_x} = I \cup \{p'\} = D_x$. Thus, $P(k+1)$ is true in this case.

- $R_{D_x} \subset R_I$: If $\mathrm{bd}(B(p', r^{p'}_{\min}))$ does not intersect with $\mathrm{bd}(R_I)$, set $y_x = p'$. Since $y_x \in R_I$, we get $D_{y_x} = I \cup \{p'\} = D_x$. Thus, $P(k+1)$ is true in this case.

  Otherwise, $\mathrm{bd}(B(p', r^{p'}_{\min}))$ intersects $\mathrm{bd}(R_I)$ at the boundary of some disk whose center is in I i.e., $\exists p \in I$ such that $y_x = \mathrm{bd}(B(p', r^{p'}_{\min})) \cap \mathrm{bd}(B(p, r^p_{\min})) \in R_{D_x}$. Since y_x intersects R_I and $B(p', r^{p'}_{\min})$, $D_y = D_x$. Thus, $P(k+1)$ is true in this case.

□

The following result shows that the sizes of the optimal solutions to `SetCover-D` and `SetCover` are equal. This property enables an improved approximation factor over [124].

Lemma 17. For any instance $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$ of Sample Placement where $P = \{p_1, \dots, p_n\}$ is finite, we have $|J^*(I)| = |S^*_{r_{\min}}(I)|$.

Proof. We have that $|J^*(I)| \geq |S^*_{r_{\min}}(I)|$ since the feasible solutions resulting from `SetCover` are a subset of the feasible solutions for `SetCover-D`. For the sake of contradiction, suppose $|J^*(I)| > |S^*_{r_{\min}}(I)|$. We will construct a feasible solution for `SetCover`$(P, \mathcal{D}(I))$ with size strictly smaller than $|J^*(I)|$. Consider the optimal solution $S^*_{r_{\min}}(I)$. For each $x \in S^*_{r_{\min}}(I)$, let $D_x := \{p \in P : x \in B(p, r^p_{\min})\}$. Then, by Lemma 16, there exists $D_{y_x} \in \mathcal{D}(I)$ such that $D_x \subseteq D_{y_x}$. Since $S^*_{r_{\min}}(I)$ is feasible for `SetCover-D`$(\mathcal{B}_{r_{\min}})$, $\cup_{x \in S^*_{r_{\min}}(I)} D_x = P$ which implies the collection $\{D_{y_x} : x \in S^*_{r_{\min}}(I)\}$ is feasible for `SetCover`$(P, \mathcal{D})$ with size $|S^*_{r_{\min}}(I)| < |J^*(I)|$ contradicting optimality of $J^*(I)$. □

We now characterize the worst-case instances that will ultimately help obtain the approximation factor.

Lemma 18 (Worst Case Covers)**.** For any instance $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$ of Sample Placement where $P = \{p_1, \dots, p_n\}$ is finite,

$$\frac{|S^*_{r_{\min}}(I)|}{|S^*_{r_{\max}}(I)|} \leq n \tag{5.19}$$

with equality achieved iff the instance I satisfies the following:

C1 All points in P lie in some disk of radius $2r_{\max}$ i.e., $\exists x \in \mathbb{R}^2$ such that $\forall i \in [n]$, $p_i \in B(x, 2r_{\max})$.

C2 The disks in the set $\mathcal{B}_{r_{\min}}(I)$ are disjoint i.e., for any $i \neq j \in [n]$, $B(p_i, r_{\min}^{p_i}) \cap B(p_j, r_{\min}^{p_j}) = \emptyset$.

Moreover, if an instance I satisfies C1 and C2, then

$$\frac{|S^*_{r_{\min}}(I)|}{|S^*_{r_{\max}}(I)|} \leq \left(\frac{r_{\min}^{[n]} + r_{\max}}{r_{\min}^{[1]}} \right)^2 . \tag{5.20}$$

Proof. Since the optimal solution to `SetCover-D` must use at least one disk and at most n disks, we obtain $|S^*_{r_{\min}}(I)| \leq n$ and $|S^*_{r_{\max}}(I)| \geq 1$ implying (5.19). Furthermore, equality is achieved when $|S^*_{r_{\min}}(I)| = n$ and $|S^*_{r_{\max}}(I)| = 1$. We will first prove the forward direction i.e., if equality is achieved, conditions C1 and C2 are satisfied.

$\Rightarrow$ Suppose C1 is not satisfied. Then, there exists a pair of disks whose centers are separated by a distance strictly greater than $2r_{\max}$. The optimal solution will then require at least two measurements which contradicts $|S^*_{r_{\max}}(I)| = 1$.

Suppose C2 is not satisfied. Then, there exists a point that belongs to at least two disks (out of n) in $\mathcal{B}_{r_{\min}}(I)$. Then, the optimal solution will have size strictly less than n which contradicts $|S^*_{r_{\min}}(I)| = n$.

$\Leftarrow$ If C1 is satisfied, the set $\{x\} \in \mathcal{X}$ is an optimal solution to `SetCover-D`($\mathcal{B}_{r_{\max}}$) (since it is within $r_{\max}$ of each point in P). If C2 is satisfied, the size of the optimal solution is at least the number of disjoint disks which is n i.e., $|S^*_{r_{\min}}| = n$.

To show (5.20), consider any instance I that satisfies conditions C1 and C2. The test points P of the instance lie within a disk of radius $2r_{\max}$ and the disks $\mathcal{B}_{r_{\min}}(I)$ are disjoint. Using areas of the smallest disk of radius $r_{\min}^{[1]}$ and the enclosing disk of radius $r_{\max} + r_{\min}^{[n]}$, we get

$$|S^*_{r_{\min}}(I)| \; \pi \left(r_{\min}^{[1]} \right)^2 \leq |S^*_{r_{\max}}(I)| \; \pi \left(r_{\max} + r_{\min}^{[n]} \right)^2 , \tag{5.21}$$

which gives the final result. $\square$

We are now ready to give the proof of the approximation factor for IntersectCover.

Proof of Theorem 9. Given an instance I, let $S(I)$ be the solution produced by IntersectCover and $S^*(I)$ be the optimal solution to Sample Placement. First, using a β-approximation algorithm for `SetCover` and Lemma 17 gives

$$|S(I)| \leq \beta |J^*(I)| = \beta |S^*_{r_{\min}}(I)|. \tag{5.22}$$

Next, using Lemma 18, we have

$$|S^*_{r_{\min}}(I)| \leq \left(\frac{r_{\max} + r_{\min}^{[n]}}{r_{\min}^{[1]}} \right)^2 |S^*_{r_{\max}}(I)| \tag{5.23}$$

Since $S^*(I)$ is a $r_{\max}$-cover of P (Lemma 11 with $\mathcal{X} = P$),

$$|S^*_{r_{\max}}(I)| \leq |S^*(I)|. \tag{5.24}$$

Combining (5.22), (5.23), (5.24) and $r_{\max} > r_{\min}^{[n]}$ gives the result. □

Algorithm 6: TSPNTour

```
   Input: Instance of Sample Placement $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$
   Output: Tour $T \subset \mathcal{X}$
1  For $i \in [n]$, compute $r_{\min}^{p_i}$ according to Section 5.3.1
2  $\mathcal{B} = \{B(p_i, r_{\min}^{p_i}) : i \in [n]\}$
3  Compute a tour $T$ for TSPN-D($\mathcal{B}$) using [42]
4  return $T$
```

TSPNTour We give an approximation algorithm in Algorithm 6 for Shortest Tour that uses the algorithm in [42]. The idea is simple: find a tour of minimum length that visits the disks centered at P with radii given by the sufficient condition in Section 5.3.1. This ensures feasibility and also yields a worst-case guarantee.

Theorem 10. Define $\omega := \frac{r_{\max}}{r_{\min}^{[1]}}$. For a finite test set P, TSPNTour is an $O(\omega)$-approximation for Shortest Tour.

Remark (Comparison with [124])**.** The work in [124] gives an $O(\frac{r_{\min}^{[n]}}{r_{\min}^{[1]}})$-approximation for `TSPN-D`. In contrast, we give the first $O(\frac{r_{\max}}{r_{\min}^{[1]}})$-approximation for Shortest Tour.

First, we give a lemma on covering a set of maximally independent disks.

Lemma 19. Let $P = \{p_1, \ldots, p_n\} \in \mathbb{R}^2$ be a finite set and let $M \subseteq P$ be a maximal $2r_{\max}$-packing of P. For each point $p \in M$, let $S_p \subset \mathbb{R}^2$ be an $r_{\min}^{[1]}$-covering of $B(p, 2r_{\max})$. Then, $\cup_{p \in M} S_p$ is a $r_{\min}^{[1]}$-covering of P.

Proof. Since M is a maximal $2r_{\max}$-packing of P, M is also a $2r_{\max}$-covering of P. Thus, for each $p \in P$, there exists $p' \in M$ such that $p \in B(p', 2r_{\max})$. Since for each $p' \in M$, $S_{p'}$ is a $r_{\min}^{[1]}$-covering of $B(p', 2r_{\max})$, we get the final result. $\square$

We present the proof of the approximation factor for TSPNTour.

Proof of Theorem 10. First, we will denote a few key quantities. Let T be the tour produced by TSPNTour and T^* denote the optimal solution of Shortest Tour. Let $M \subset P$ be a maximal $2r_{\max}$-packing of P and define $\mathcal{B}_M := \{B(p, r_{\max}) : p \in M\}$. Recall that $\mathcal{B}_{r_{\min}}(I) := \{B(p_i, r_{\min}^{p_i}) : i \in [n]\}$. Let T_M^* and T_r^* denote the optimal solution of `TSPN-D`$(\mathcal{B}_M)$ and `TSPN-D`$(\mathcal{B}_{r_{\min}})$ respectively.

Since T^* is a $r_{\max}$-covering of P, T^* is a feasible tour for `TSPN-D`$(\mathcal{B}_M)$. Then,

$$\text{length}(T_M^*) \leq \text{length}(T^*) \tag{5.25}$$

Note that we take $|T_M^*| = |M|$. If $|T_M^*| > |M|$, then there exists a vertex $v \in T_M^*$ such that either v does not visit any disk in $\mathcal{B}_M$ or v visits a disk that has already been visited on the tour T_M^* (pigeonhole principle). In either case, by removing the vertex v and using the triangle inequality, we obtain a feasible tour of length equal to or shorter than length(T_M^*).

We now construct a feasible tour for `TSPN-D`$(\mathcal{B}_{r_{\min}})$ from T_M^*. For each $p \in M$, consider the disk $B(p, 2r_{\max})$. A hexagonal tour T_{HC} of length $O(\frac{r_{\max}^2}{r_{\min}})$ exists whose vertex set is a $r_{\min}^{[1]}$-covering of $B(p, 2r_{\max})$ (Lemma 13 with $\mathcal{X} = B(p, 2r_{\max})$ and $r_{\min} = r_{\min}^{[1]}$). Then, for each vertex $v \in T_M^*$, a detour of length $2r_{\min}^{[1]} + \text{length}(T_{\text{HC}})$ guarantees feasibility for Shortest Tour (Lemma 19). The length of this tour is atleast the length of the optimal tour T_r^* i.e.,

$$\text{length}(T_r^*) \leq \text{length}(T_M^*) + |M| \left(2r_{\min}^{[1]} + O\left(\frac{r_{\max}^2}{r_{\min}^{[1]}} \right) \right). \tag{5.26}$$

Now, the length of any tour of $|M|$ disjoint disks of radius $r_{\max}$ is at least $0.239|M|r_{\max}$ [123, Theorem 1] i.e.,

$$|M| \leq \frac{4.2}{r_{\max}} \text{length}(T^*_M). \tag{5.27}$$

Combining (5.25), (5.26), (5.27) and using $r^{[1]}_{\min} < r_{\max}$ gives

$$\frac{\text{length}(T^*_r)}{\text{length}(T^*)} = O\left(\frac{r_{\max}}{r^{[1]}_{\min}}\right). \tag{5.28}$$

Since the algorithm [42] is an $O(1)$-approximation i.e., $\text{length}(T) \leq c\ \text{length}(T^*_r)$ for some constant $c > 1$, we obtain the final result. □

The major pitfall with TSPNTour is that the algorithm in [42] has a large running time of at least $\Omega(n^4)$ (see [42] for a precise characterization). As a result, we provide a natural heuristic that is easy to implement in the next section.

IntersectTour We describe a natural heuristic for Shortest Tour. Since we already have a feasible solution using IntersectCover, we plan an approximate tour on this set. The steps are outlined in Algorithm 7. If we use Christofides' algorithm, the running time of IntersectTour is $O(n^3)$. In practice, one will either use optimal solvers or heuristics [66]. While we have not provided a performance guarantee, this heuristic is simple to implement and yields good results in practice.

```
Algorithm 7: IntersectTour
  Input: Instance of Sample Placement $I = (\mathcal{X}, \rho, P, k, w, r_{\max})$
  Output: Tour $T \subset \mathcal{X}$
1 $S$ = IntersectCover($I$)
2 $T$ = ApproximateTour($S$)
3 return $T$
```

Remark (Comparison with [124]). The work in [124] proposes a heuristic, which we refer to by GridCoverTour, that discretizes the environment into a grid, computes a minimum set cover, and plans a tour on the selected points. The key difference between GridCoverTour and IntersectTour is how the subsets for `SetCover` are generated: grids versus intersection points. The grid discretization in $\mathbb{R}^2$ is determined by a resolution parameter

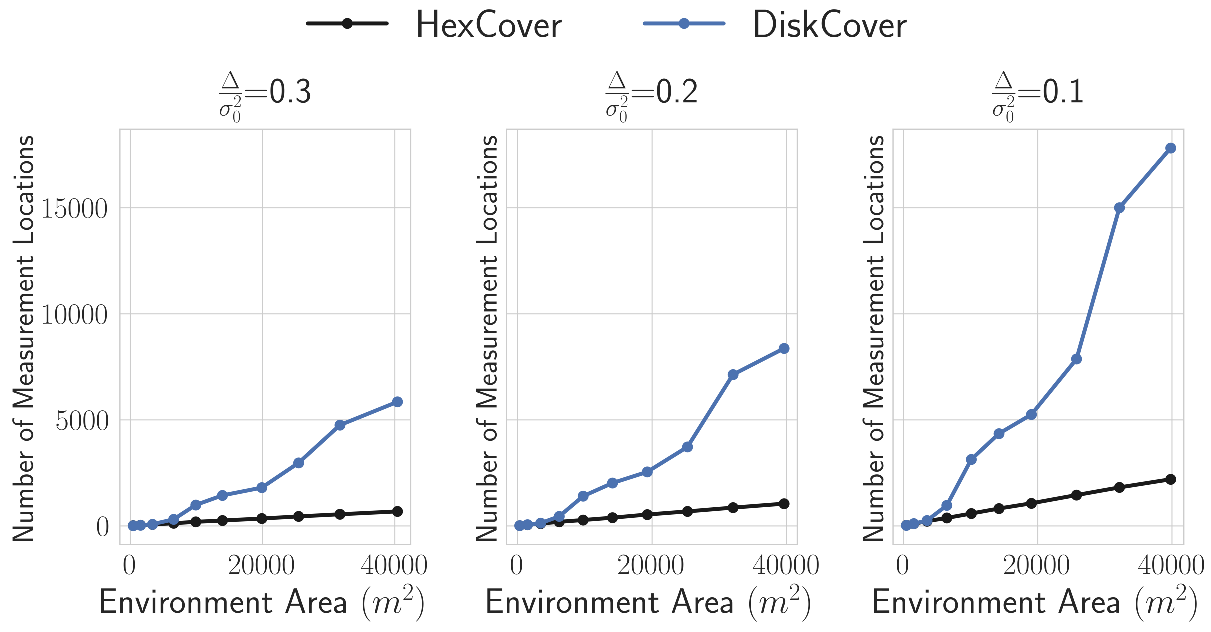


Figure 5.4: Comparing solution size for SAMPLE PLACEMENT in convex environments versus environment area. DISKCOVER uses roughly 6 times as many measurements as HEXCOVER.

$\theta \geq 1$ resulting in $O(\theta^2)$ subsets compared to $O(\gamma n)$ subsets for INTERSECTTOUR. Further, there is no guarantee that a grid discretization preserves optimality for `SetCover-D` unlike our approach (Lemma 17).

5.4 Numerical Results

This section covers simulation results comparing our proposed algorithms against prior work [122, 124]. We begin by describing the experimental setup. The experiments are run on an AMD Ryzen 7 2700 processor with 16GB of RAM.

Experimental Setup

We follow the setup in [122] where a Gaussian Process was fit to a real world dataset of organic matter measurements in an agricultural field. The authors computed $L =$

8.33 meters, $\sigma_0 = 12.87$, and $\sigma^2 = 0.0361$. Since the covariance function is isotropic, only the relative distances between points matter. This enables us to consider different environment sizes similar to the setup in [44]. We consider rectangular environments whose area ranges from 400 to 40000 square metres. Further, for convex environments, we consider three regimes of desired accuracy: $\Delta/\sigma_0^2 = 0.3, 0.2, 0.1$. These correspond to keeping the posterior variance under 30%, 20%, and 10% of the initial value σ_0^2, respectively. For finite test sets, we consider three regimes: sparse (30 test points), moderate (50 test points), and dense (100 test points). We sample the threshold variances uniformly at random in the range $[0.1 \times \sigma_0^2, 0.5 \times \sigma_0^2]$. For each configuration (test point regime and threshold variances), we generate 10 random samples and report the results. To compute approximate covers, we use the greedy algorithm [49].

5.4.1 Convex Environments, Constant Thresholds

In this section, we demonstrate the effectiveness of HexCover, HexCoverTour over DiskCover, DiskCoverTour [122] across environments of different sizes.

Remark. The DiskCover and DiskCoverTour algorithms end up placing samples outside the environment, which is not practically possible. In the simulations considered, we remove these samples and the redundant measurement locations that arise (as mentioned in the paper [122]). In addition, for DiskCoverTour, we use approximation algorithms for the TSP instead of the proposed lawn-mower tours. This change leads to shorter tour lengths in practice.

The results for Sample Placement are shown in Figure 5.4. In each subplot, the number of measurements is plotted against the environment size. The difference across the subplots is the error tolerance. As the variance threshold decreases, the number of measurements required will increase for any algorithm. In each subplot, we observe that DiskCover uses more measurements than HexCover (ours). In small environments, the difference is negligible as it does not take many measurements to get a feasible solution. However, as the environment size increases, DiskCover uses roughly 6 times as many measurements as HexCover. Further, the number of measurements used by HexCover increases roughly linearly with the environment size. This is not the case for DiskCover whose measurement set size grows much faster.

The results for Shortest Tour are shown in Figure 5.5. In each subplot, the tour length is plotted against the environment size. As before, the difference across the subplots is the variance threshold. As the threshold decreases, the number of samples required

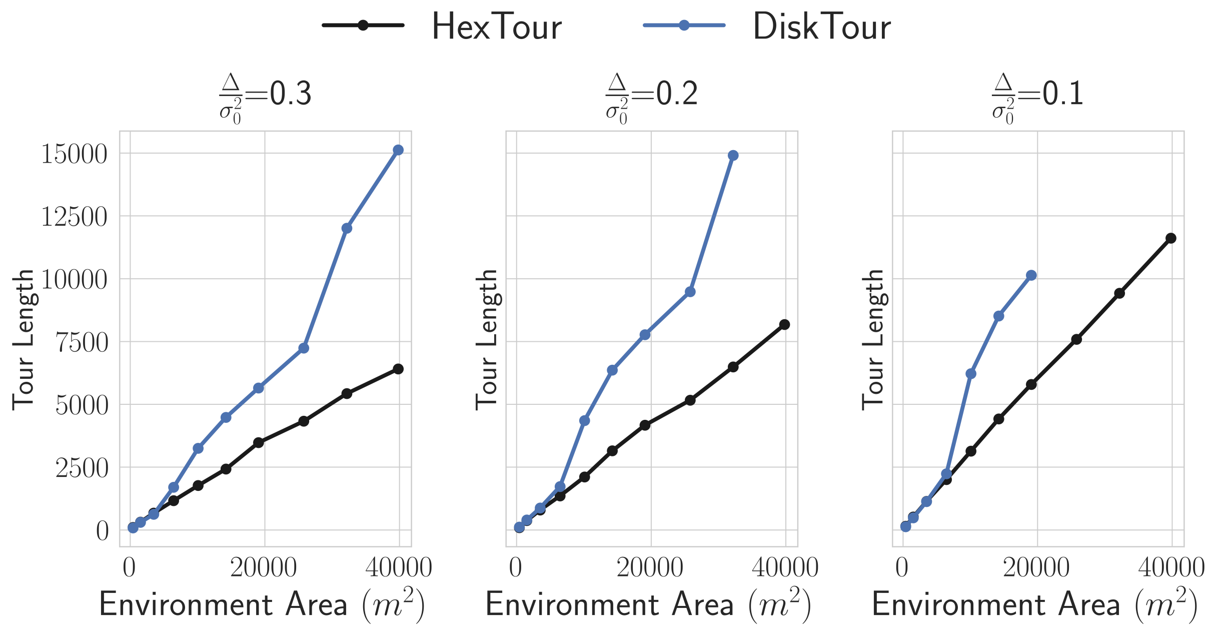
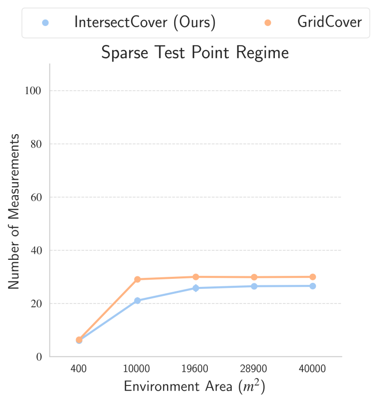
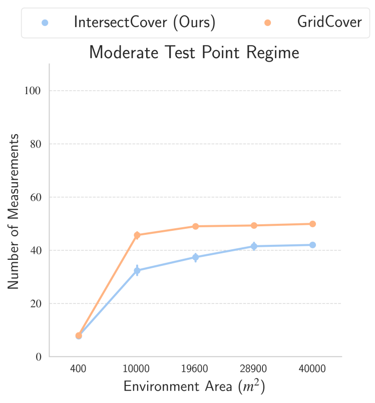


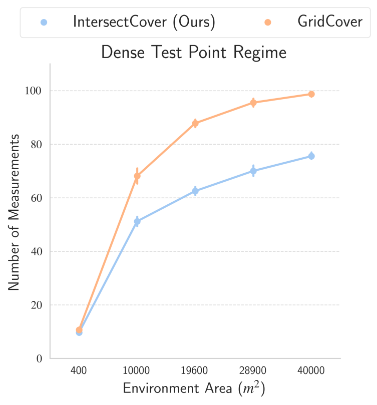
Figure 5.5: Tour lengths for SHORTEST TOUR in convex environments versus environment area. HEXCOVERTOUR plans tours of length that are roughly half that of DISKCOVERTOUR.



(a)



(b)



(c)

Figure 5.6: Comparing solution size for Sample Placement for finite test sets versus environment area. GridCover uses roughly 1.25 times as many measurements as IntersectCover in the moderate and dense test point regimes.

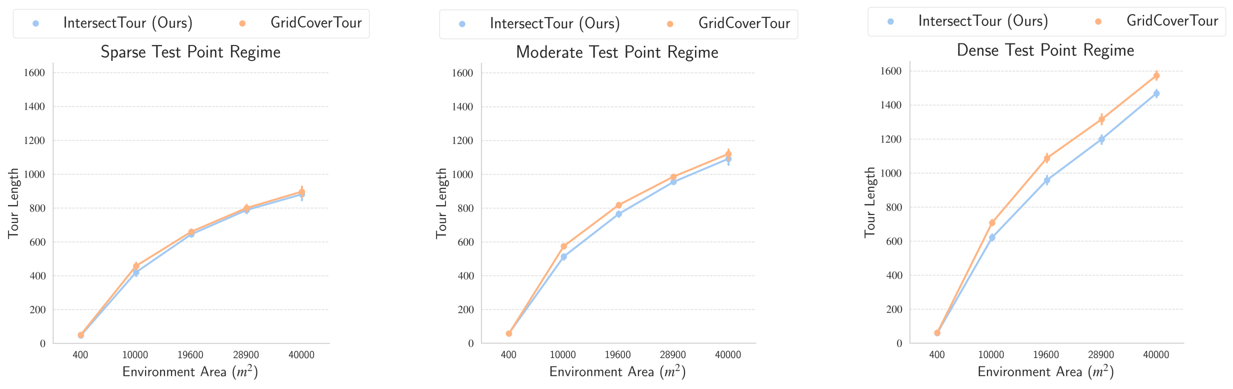
will increase, which subsequently increases the tour length. In each subplot, we see that DiskCoverTour produces tours of length larger than HexCoverTour (ours). For small environments, the difference in tour length is negligible. However, for larger environments, HexCoverTour produces tours that are roughly half the length of tours produced by DiskCoverTour. Further, in the second and third plots, one may notice missing data points for DiskCoverTour. This is because the number of vertices become too large ($\geq$ 7500) to run Christofides' algorithm in a reasonable amount of time and memory. In contrast, HexCoverTour scales better with the environment size.

5.4.2 Finite Test Sets, Arbitrary Thresholds

In this section, we demonstrate the effectiveness of IntersectCover, IntersectTour over GridCover, GridCoverTour [124] across environments of different sizes.

Remark. Since the authors in [124] do not mention a specific grid resolution to ensure good quality solutions, we choose a resolution that ensures the computation times are comparable to our proposed algorithms. We accomplish this by setting the number of grid points equal to ceil(m) where m are the total number of subsets computed by GenSubset. Then, the greedy algorithm for set cover will have roughly the same runtime for both algorithms.

The results for Sample Placement are shown in Figure 5.6. Across all test point



(a) (b) (c)

Figure 5.7: Tour lengths for Shortest Tour for finite test sets versus environment area. GridCoverTour plans tours of length that are longer than IntersectTour across all regimes with the largest difference seen in the dense regime.

regimes, IntersectCover uses fewer measurements than GridCover. Specifically, GridCover uses roughly 1.25 times the number of measurements used by IntersectCover in the moderate and dense regimes. An interesting observation here is that as the environment area increases, GridCover requires as many measurements as the number of test points (30 in sparse, 50 in moderate, 100 in dense). In contrast, IntersectCover uses fewer measurements (26 in sparse, 40 in moderate, 80 in dense). Thus, by using the intersections of the disks, IntersectCover uses fewer measurements compared to a grid discretization which is oblivious to the spatial distribution of test points.

The results for Shortest Tour are shown in Figure 5.7. Across all regimes, IntersectTour computes shorter tours compared to GridCoverTour. The performance difference is greatest in the dense regime where GridCoverTour produces tours that are roughly 1.15 times the tour lengths given by IntersectTour. In the sparse and moderate regime, the performance difference is negligible since any tour would have to visit at least as many test points leading to similar results. Unless a tour can visit significantly fewer locations than the number of test points, we should expect similar tour lengths across both algorithms. We expect significantly shorter tour lengths for larger environments. For example, agricultural fields in practice are a magnitude larger than the environments considered in these experiments [129].

5.5 Summary

In this chapter, we studied the following version of informative path planning: minimize resources used (samples or tour length) subject to constraints on the posterior variance. We gave approximation algorithms for convex environments and finite test sets. We also gave a natural heuristic that yielded good solutions in practice. Our approach was rooted in finding minimum size covers of the test points.

Chapter 6

Subset Selection in Random Fields via Maximal Cliques

6.1 Introduction

An important problem in engineering applications is deciding the subset of measurements that are the most useful in the estimation of an unknown quantity of interest. For example, in agriculture, it is important to estimate the nutrient quality of a field using soil samples. This helps guide fertilizer usage to replenish lost nutrients, which subsequently maximizes crop yield. However, it is impractical to sample the soil at each location in large agricultural fields. The goal is to determine where to sample the soil, such that the nutrient quality at a large set of prediction locations can be estimated accurately. An example of the soil pH variability in a field with a set of prediction locations is shown in Figure 6.1. This type of subset selection problem shows up in domains such as sensor placement/active sampling in spatial statistics [78, 108, 82, 85, 137], feature selection in machine learning [92, 61], informative path planning in robotics [12, 10, 11, 122], among others. The challenge is similar: choose the subset of attributes that best estimates the quantity of interest.

The Bayesian approach is to model the quantities as random variables. The estimates of prediction variables are obtained by linear estimators and the quality of the chosen subset is measured by the resulting mean squared estimation error. The benefits of this approach are twofold. First, prior statistical knowledge of the quantities can be incorporated into the estimation procedure. Second, the mean squared error resulting from a linear estimator is independent of the observations. Thus, deciding the subset that minimizes the mean squared error can be done *a priori*. With a finite observation set, a popular

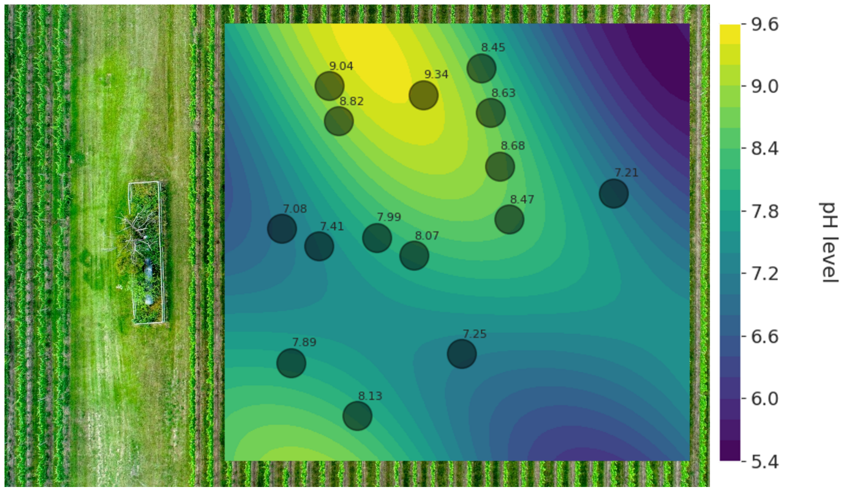


Figure 6.1: An example of the pH variability in an agricultural field. The circles are the prediction locations where accurate estimates are desired. Agricultural fields can be large and one can only take a fixed number of soil samples to best estimate the pH variability at the prediction locations.

approach is to use the greedy algorithm [92, 64]. Each step, the variable maximizing the marginal gain is selected. Continuous observation sets, such as agricultural fields, can be made finite by a grid discretization, which can be used by the greedy algorithm, which we refer to as Grid-Greedy. The solution quality improves with a finer grid but at an increased computational cost. The objective of the work in this chapter is to remove the dependence on the grid discretization while obtaining good solution quality. Our proposed method, Centroid-Greedy, restricts the search to the set of prediction locations and the centroids of the cliques formed by the prediction locations. This is motivated by our analysis of the problem in one dimension where we identify a critical distance between points that characterize the optimal measurement location. In our experiments, we show Centroid-Greedy achieves better solutions when given the same computational resources as Grid-Greedy and finds solutions of similar quality more efficiently.

6.1.1 Contributions

The contributions of this chapter are twofold. First, we formulate a problem of budget constrained observation selection from an infinite set to best estimate a finite set of prediction variables. Second, we propose Centroid-Greedy, a greedy algorithm that uses a ground set consisting of the prediction locations and the centroids of cliques formed by the prediction locations. This reduces the dependence of Grid-Greedy on the grid discretization of the continuous field. In simulations, we demonstrate the improved solution quality and run time of Centroid-Greedy in comparison to Grid-Greedy.

6.2 Problem Formulation

Let $D \subset \mathbb{R}^d$ represent the environment and let $\sigma_0 \in \mathbb{R}_{>0}$ be a positive real number. For any location $x \in D$, let $Z(x)$ be a random variable with zero mean and variance σ_0^2. We consider a convex set of measurement locations $\Theta \subset D$ and a finite set of prediction locations $\Omega \subset \Theta$. Given a positive integer $k \in \mathbb{Z}_+$, our goal is to minimize the mean-squared error of the linear estimation of the prediction variables $\{Z(x) : x \in \Omega\}$ using only k measurement variables.

Remark. Note that when $|\Omega| < k$, measurements at all prediction locations will yield low estimation error. The problem is only interesting when $|\Omega| > k$.

We define $\phi_{\mathrm{SE}} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ to be the squared exponential covariance function with known parameters σ_0 and $L \in \mathbb{R}_{>0}$:

$$\phi_{\mathrm{SE}}(x) = \sigma_0^2 e^{-\frac{x^2}{2L^2}}, \tag{6.1}$$

The parameters can be learned from a pilot deployment or expert knowledge and is a standard assumption in sensor placement algorithms [78].

For any $x, y \in D$, we assume the covariance of the random variables $Z(x), Z(y)$ is given by

$$\mathrm{Cov}(Z(x), Z(y)) = \mathbb{E}[Z(x)Z(y)] = \phi_{\mathrm{SE}}(\|x - y\|). \tag{6.2}$$

Let i be a positive integer. For any $x \in D$, let $Y_i(x)$ be the i^{th} noisy measurement of $Z(x)$ and let the associated noise be $\epsilon_i(x)$. The noise is assumed to be a zero mean random variable with variance $\sigma^2 > 0$. In addition, the noise is uncorrelated across measurements and locations i.e. for any $x, y \in D$ and for any positive integers $m, n \in \mathbb{Z}_+$, $\mathrm{Cov}(\epsilon_m(x), \epsilon_n(y)) = 0$. The measurement is

$$Y_i(x) = Z(x) + \epsilon_i(x). \tag{6.3}$$

In order to reduce notational clutter, for any $x \in D$, we drop the subscript i from the measurement variable $Y_i(x)$ and the associated noise $\epsilon_i(x)$. We proceed with the understanding that if there are multiple measurements at the same location, the associated noise terms are uncorrelated. In addition, any measurement at the same location $x \in D$ (even if there are multiple) will be denoted by $Y(x)$. Now, the measurement equation is

$$Y(x) = Z(x) + \epsilon(x). \tag{6.4}$$

We wish to minimize the total mean-squared error, which gives us the following constrained optimization problem:

$$\min_{S \subset \Theta, |S| \leq k} \sum_{x \in \Omega} \mathbb{E}\left[\left(Z(x) - \hat{Z}(x,S)\right)^2\right], \tag{6.5}$$

where $\hat{Z}(x,S)$ is the linear estimator of $Z(x)$ given the variables in S. We will now rewrite the problem using the definition of the mean squared error. Denote the elements of a set S by $\{x_1, \ldots, x_k\}$. The linear estimator $\hat{Z}(x,S)$ is given by Theorem 1:

$$\hat{Z}(x,S) := \boldsymbol{b}_x(S)^T C(S)^{-1} \boldsymbol{Y}_S \tag{6.6}$$

where

$$\begin{aligned} \boldsymbol{b}_x(S) &:= [\phi_{\mathrm{SE}}(\|x - x_1\|), \ldots, \phi_{\mathrm{SE}}(\|x - x_k\|)] \in \mathbb{R}^k \\ \boldsymbol{Y}_S &:= [Y(x_1), \ldots, Y(x_k)] \in \mathbb{R}^k \\ C(S) &:= \mathbb{E}\left[\boldsymbol{Z}_S \boldsymbol{Z}_S^T\right] + \sigma^2 I_k \in \mathbb{R}^{k \times k} \\ &= \begin{bmatrix} \phi_{\mathrm{SE}}(0) & \ldots & \phi_{\mathrm{SE}}(\|x_1 - x_k\|) \\ \vdots & \ddots & \vdots \\ \phi_{\mathrm{SE}}(\|x_k - x_1\|) & \ldots & \phi_{\mathrm{SE}}(0) \end{bmatrix} + \sigma^2 I_k. \end{aligned} \tag{6.7}$$

Expanding (6.5), we get

$$\min_{S \subset \Theta, |S| \leq k} \sum_{x \in \Omega} \phi_{\mathrm{SE}}(0) - \boldsymbol{b}_x(S)^T C(S)^{-1} \boldsymbol{b}_x(S). \tag{6.8}$$

Since $\phi_{\mathrm{SE}}(0) = \sigma_0^2$ is a constant, we can consider the maximization version of the problem. Define

$$\begin{aligned} f_x(S) &:= \boldsymbol{b}_x(S)^T C(S)^{-1} \boldsymbol{b}_x(S) \\ f(S) &:= \sum_{x \in \Omega} f_x(S), \end{aligned} \tag{6.9}$$

where $\boldsymbol{b}_x(S)$ and $C(S)$ are defined in (6.7). The function $f_x(S)$ is also known as the *squared multiple correlation* [92, 36] or the *variance reduction* [77].

In this chapter, we wish to find the measurement set that maximizes the total variance reduction. This is formulated as the following optimization problem.

Problem 8. Given measurement locations Θ, prediction locations Ω, and a budget $k > 0$, find a measurement set $S \subset \Theta$ of size k that maximizes the total variance reduction:

$$\max_{S \subset \Theta, |S| \leq k} f(S). \tag{6.10}$$

6.3 Problem Structure

In this section we provide preliminary results that guide the design of our algorithm, presented in the next section.

Non-submodularity Problem 1 resembles a sensor placement problem where one is interested in a subset of locations to deploy sensors to maximize the information gained about the environment. Metrics related to the information gained such as coverage and mutual information are known to be submodular functions which can be approximately solved efficiently with a guarantee. However, for Problem 1, we provide an example to show the variance reduction objective is *not submodular*.

Example 2. Consider the following environment setup where the points lie on an interval on the real line.

$$\begin{aligned} &D \subset \mathbb{R}, \Omega = \{0\}, \Theta = [0,2], \sigma = 1, \sigma_0 = 1, L = 1, \\ &A = \{0.6784\}, B = \{0.6784, 1.4869\}, x = 0.6892. \end{aligned} \tag{6.11}$$

Now, $f(A \cup \{x\}) - f(A) = 0.1021$ and $f(B \cup \{x\}) - f(B) = 0.1025$, which shows the violation. $\square$

Two Prediction Locations with One Sample We discuss properties of the problem in 1-D, i.e., the random variables are associated with locations on the real line. This restriction provides valuable insight into the problem and motivates our proposed algorithm.

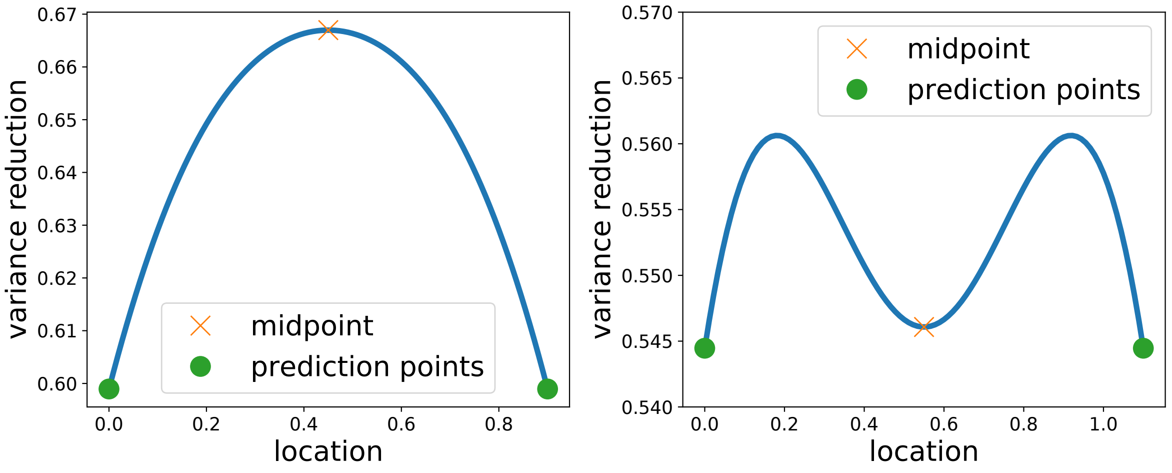


Figure 6.2: Left: An example of the objective function defined on the interval $[0, 1]$. The test locations are located at $y_1 = 0.0, y_2 = 0.9$ and $L = \frac{1}{\sqrt{2}}$. In this setting, the midpoint achieves the global maximum. Right: An example of the objective function defined on the interval $[0, 1.1]$. The test locations are located at $y_1 = 0.0, y_2 = 1.1$ and $L = \frac{1}{\sqrt{2}}$. In this setting, the midpoint is a local minima.

Suppose the set of prediction locations contains two points i.e. $\Omega = \{y_1, y_2\} \subset [a, b]$, with $y_1 < y_2$. After some simplification, the optimization problem in (6.10) is

$$\begin{aligned} &\max_{x\in[a,b]} \frac{1}{\sigma_0^2 + \sigma^2} \left(\phi_{\text{SE}}^2(\|x - y_1\|) + \phi_{\text{SE}}^2(\|x - y_2\|)\right) \\ =&\max_{x\in[a,b]} \frac{\sigma_0^4}{\sigma_0^2 + \sigma^2} \left(e^{-\frac{1}{L^2}\|x-y_1\|^2} + e^{-\frac{1}{L^2}\|x-y_2\|^2}\right). \end{aligned} \tag{6.12}$$

The solution depends on the relationship between the distance between the two prediction locations and L, the parameter of the squared exponential covariance function defined in (6.1). This is formalized in the following proposition.

Proposition 1. Let $D \subset \mathbb{R}$, $\Omega = \{y_1, y_2\} \subset D, \Theta = D$, $t = 1$, and the midpoint $x^* = \frac{y_1+y_2}{2}$. Denote the optimal solution to (6.12) by OPT. Then,

$$\text{OPT} = x^* \iff \|y_2 - y_1\| \leq \sqrt{2}L. \tag{6.13}$$

Proof. Define $d_1 := \|x - y_1\|, d_2 := \|x - y_2\|$. In this setting, the objective function is

$$f(x) = \frac{\sigma_0^4}{\sigma_0^2 + \sigma^2} \left(e^{-\frac{d_1^2}{L^2}} + e^{-\frac{d_2^2}{L^2}}\right). \tag{6.14}$$

The derivative of $f(x)$ is:

$$f'(x) = \frac{-2}{L^2}\frac{\sigma_0^4}{\sigma_0^2+\sigma^2}\left((x-y_1)e^{-\frac{d_1^2}{L^2}} + (x-y_2)e^{-\frac{d_2^2}{L}}\right). \tag{6.15}$$

In general, it is difficult to solve for the critical points using the derivative of the form in (6.15) since it is a transcendental equation. One could resort to numerical methods to solve it. However, in this case, we identify the midpoint $x^* = \frac{y_1+y_2}{2}$ as a critical point for the function i.e. $f'(x^*) = 0$.

The second derivative is given by:

$$\begin{aligned} f''(x) = \frac{-2}{L^2}\frac{\sigma_0^4}{\sigma_0^2+\sigma^2}\Bigg(& e^{-\frac{d_1^2}{L^2}}\big(1-\frac{2}{L^2}(x-y_1)^2\big) \\ & +e^{-\frac{d_2^2}{L^2}}\big(1-\frac{2}{L^2}(x-y_2)^2\big)\Bigg) \end{aligned} \tag{6.16}$$

Evaluating the second derivative at the critical point $x^* = \frac{y_1+y_2}{2}$ gives

$$\begin{aligned} f''(x^*) = \frac{-4}{L^2}\frac{\sigma_0^4}{\sigma_0^2+\sigma^2}& e^{-\frac{1}{4L^2}(y_2-y_1)^2} \\ & \left(1-\frac{1}{2L^2}(y_2-y_1)^2\right) \end{aligned} \tag{6.17}$$

$\Rightarrow$ We will prove the forward direction by proving the contrapositive. When $(y_2-y_1)^2 > 2L^2$, $f''(x^*) > 0$ and x^* is a local minima and is not optimal.

$\Leftarrow$ Since $(y_2-y_1)^2 < 2L^2$, we have that $f''(x^*) < 0$, and thus x^* is a local maxima. To show x^* is a global maximum, we show $f'(x) > 0$ on the interval $[y_1, x^*]$ and $f'(x) < 0$ on the interval $[x^*, y_2]$.

On the interval $[y_1, x^*]$, it is sufficient to show the following:

$$\frac{x-y_1}{y_2-x} \le \frac{e^{-\frac{d_2^2}{L^2}}}{e^{-\frac{d_1^2}{L^2}}}, \tag{6.18}$$

since this ensures $f'(x) > 0$. Consider the RHS in (6.18),

$$\begin{aligned} \frac{e^{-\frac{d_2^2}{L^2}}}{e^{-\frac{d_1^2}{L^2}}} &= e^{-\frac{1}{L^2}\left((x-y_2)^2-(x-y_1)^2\right)} \\ &= e^{-\frac{1}{L^2}(y_1-y_2)(2x-y_2-y_1)} = e^{-\frac{2}{L^2}(y_2-y_1)(\frac{y_1+y_2}{2}-x)}. \end{aligned} \tag{6.19}$$

Since $(y_2-y_1)^2 < 2L^2$, it holds that $-\frac{2}{L^2}(y_2-y_1) > -\frac{4}{(y_2-y_1)}$. Continuing from (6.19) gives

$$e^{-\frac{2}{L^2}(y_2-y_1)(\frac{y_1+y_2}{2}-x)} > e^{-\frac{4}{y_2-y_1}(\frac{y_1+y_2}{2}-x)}. \tag{6.20}$$

Now, we need to show that the LHS in (6.18) is less than the lower bound in (6.20). Define $A := \frac{y_1+y_2}{2}$, $B := \frac{y_2-y_1}{2}$, and for any $x \in [y_1, x^*]$, define $Z := x - A$. Starting with the LHS gives

$$\begin{aligned} \frac{x-y_1}{y_2-x} &= \frac{x-y_1+A-A}{y_2-x+A-A} = \frac{x+B-A}{-x+A+B} \\ &= \frac{Z+B}{-Z+B} = \frac{\frac{Z}{B}+1}{-\frac{Z}{B}+1}. \end{aligned} \tag{6.21}$$

Consider the function $g(y) := e^{2y}\frac{1-y}{1+y}$. The derivative is $g'(y) = -\frac{2y^2e^{2y}}{(1+y)^2}$ which shows the function is non-increasing for all $y \neq -1$. Then, since $g(0) = 1$, $g(y) \geq 1$ on the interval $(-1, 0]$. Then, we have for $y \in (-1, 0]$,

$$\frac{y+1}{-y+1} \leq e^{2y}. \tag{6.22}$$

Note that since $y_2 - y_1 \leq \sqrt{2}L$, for any $x \in [y_1, x^*]$, $-1 \leq \frac{Z}{B} \leq 0$. Setting $y = \frac{Z}{B}$ in (6.22) gives

$$\frac{\frac{Z}{B}+1}{-\frac{Z}{B}+1} \leq e^{2\frac{Z}{B}} = e^{-\frac{4}{y_2-y_1}(\frac{y_1+y_2}{2}-x)}, \tag{6.23}$$

which shows that the LHS in (6.18) is less than the lower bound in (6.20). Thus, $f'(x) > 0$ and $f(x)$ is increasing on the interval $[y_1, x^*]$. Using similar arguments, one can show $f(x)$ is decreasing on the interval $[x^*, y_2]$. Combining this with the fact x^* is a critical point and $f''(x^*) < 0$ implies x^* is the global maximum. □

When the points are separated by a distance greater than $\sqrt{2}L$, Proposition 1 guarantees the suboptimality of the midpoint (see Figure 6.2). In this case, the prediction locations are reasonable solutions whose performance guarantee is given by the following proposition.

Proposition 2. Given $D \subset \mathbb{R}$, $\Omega = \{y_1, y_2\} \subset D, \Theta = D$, and $k = 1$, when $\|y_2 - y_1\| > \sqrt{2}L$, the point $x = y_1$ is an approximate maximizer to (6.12) with a guarantee

$$\frac{f(\{y_1\})}{f(\{x^*\})} \geq 0.62, \tag{6.24}$$

where x^* is the optimal measurement location.

Proof. Define $d_1 := \|x - y_1\|, d_2 := \|x - y_2\|$. In this setting, the objective function is

$$f(x) = \frac{\sigma_0^4}{\sigma_0^2 + \sigma^2}\left(e^{-\frac{d_1^2}{L^2}} + e^{-\frac{d_2^2}{L^2}}\right). \tag{6.25}$$

The derivative of $f(x)$ is:

$$f'(x) = \frac{-2}{L^2}\frac{\sigma_0^4}{\sigma_0^2 + \sigma^2}\left((x - y_1)e^{-\frac{d_1^2}{L^2}} + (x - y_2)e^{-\frac{d_2^2}{L}}\right). \tag{6.26}$$

For $x < y_1$, $f'(x)$ is positive and for $x > y_2$, $f'(x)$ is negative. Thus, the optimal solution x^* must lie within the interval $[y_1, y_2]$. Since $\|y_1 - y_2\| > \sqrt{2}L$, $x = \frac{y_1+y_2}{2}$ is a local minima (Proposition 1). The function is symmetric around the midpoint, so we restrict our discussion to the interval $[y_1, \frac{y_1+y_2}{2}]$. A lower bound for the solution $x = y_1$ is constructed by assuming $e^{-\frac{\|y_1-y_2\|^2}{L^2}} = 0$. Thus, $f(y_1) \geq \frac{\sigma_0^2}{\sigma_0^2+\sigma^2}$. Since the optimal solution $x^* \in [y_1, \frac{y_1+y_2}{2}]$ and $\|y_1 - y_2\| \geq \sqrt{2}L$, an upper bound can be constructed: $f(x^*) < \frac{\sigma_0^2(1+e^{-0.5})}{\sigma_0^2+\sigma^2}$. Thus,

$$\frac{f(y_1)}{f(x^*)} \geq \frac{1}{1 + e^{-0.5}} \approx 0.62. \tag{6.27}$$

□

Propositions 1 and 2 motivate our algorithm design. For two prediction points and one sample in 1D, either the midpoint is optimal or either prediction point is an approximate solution. This suggests the following idea: restrict the search of the greedy algorithm to the prediction locations and the centroids of the cliques formed by the prediction locations.

6.4 Algorithms

In this section, we discuss Grid-Greedy and its limitations, our proposed algorithm Centroid-Greedy based on computing centroids of maximal cliques, and provide a reformulation of computing the marginal gains that speeds up the implementation of both greedy algorithms in practice.

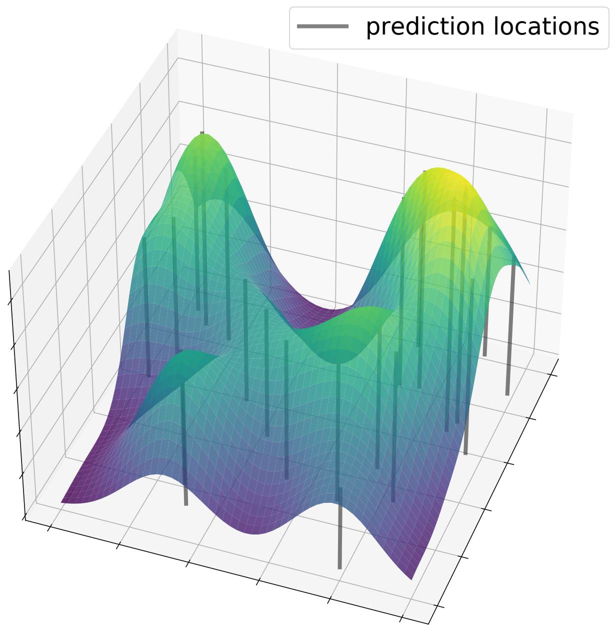


Figure 6.3: A plot of the objective $f(S)$ in Problem 1 when the random variables are associated with a two-dimensional space and the budget is one i.e. $t = 1$. The function is non-concave and has many local maxima.

6.4.1 Grid-Greedy

The greedy algorithm is popular for subset selection in regression where it is also known as Forward Selection [36, 92, 64]. In this section, we discuss how the greedy algorithm can be used for infinite observation sets. For Problem 1, starting with $S_0 = \emptyset$, the first step of the algorithm computes the maximizer to

$$\begin{aligned} S_1 &= \underset{x \in \Theta}{\arg\max}\ f(\{x\}) \\ &= \underset{x \in \Theta}{\arg\max}\ \frac{1}{\sigma^2 + \sigma_0^2} \sum_{y \in \Omega} \phi_{\text{SE}}^2(\|x - y\|) \\ &= \underset{x \in \Theta}{\arg\max}\ \frac{\sigma_0^4}{\sigma^2 + \sigma_0^2} \sum_{y \in \Omega} e^{-\frac{1}{L^2}\|x-y\|^2}. \end{aligned} \tag{6.28}$$

This function is non-concave and in general, it is difficult to find the global maximum. A plot of this objective when the set of observation locations is a subset of the two dimensional Euclidean space i.e. $\Theta \subset \mathbb{R}^2$ is shown in Figure 6.3.

To tackle this non-concave maximization problem, the set of measurement locations Θ can be uniformly discretized to form a finite set of points $\bar{\Theta} \subset \Theta$. The point $x \in \bar{\Theta}$ with the maximum function value is returned as an approximate solution. This is known as the Uniform Grid method [101]. Each step of the greedy algorithm can be approximately solved using this method. The grid discretization is determined by a positive integer parameter $\rho \geq 1$ which tiles each dimension with ρ points to form $\bar{\Theta}$ of size ρ^d. We refer to this method as GRID-GREEDY. The time complexity of GRID-GREEDY is given in the following proposition.

Proposition 3. Given a positive integer $\rho \geq 1$ and a grid discretization of size ρ^d, GRID-GREEDY finds a solution to Problem 1 in time $O(\rho^d k^3 \max\{k, |\Omega|\})$.

Proof. GRID-GREEDY runs for k iterations with ρ^d function evaluations per iteration. For a set S of size k, the evaluation of $f(S)$ requires the inversion of a $k \times k$ matrix and $|\Omega|$ matrix multiplications, with each multiplication taking time $O(k^2)$. Thus, the overall time complexity for the evaluation of $f(S)$ is $O(\max\{k^3, |\Omega|k^2\})$. Then, GRID-GREEDY runs in time $O(\rho^d k \max\{k^3, |\Omega|k^2\}) = O(\rho^d k^3 \max\{k, |\Omega|\})$. □

The dependence of the runtime on ρ^d is concerning. To get good quality solutions using the greedy algorithm, ρ needs to be sufficiently large to achieve a good grid resolution. In this chapter, we aim to find good quality solutions using the greedy algorithm in time *independent* of the grid discretization.

6.4.2 Centroid-Greedy

In this section, we present our algorithm CENTROID-GREEDY which involves two parts. First, we find the centroids of maximal cliques in a graph with nodes as prediction locations. Second, we use the set of centroids and prediction locations as a ground set for the greedy algorithm for maximizing set functions to solve Problem 1.

Finding Clique Centroids

The steps to compute clique centroids is given in Algorithm 8. The first step (Line 1, CONSTRUCTGRAPH) constructs a graph $G = (V, E)$ with vertices as prediction locations.

```
Algorithm 8: MaximalCliqueCentroids
  Input: Prediction locations $\Omega$
  Output: Clique Centroids $\mathcal{X} \subset \Theta$
1 $G = (V, E) \leftarrow$ constructGraph($\Omega$)
2 $\mathcal{C} \leftarrow$ maximalCliques($G$)
3 Initialize $\mathcal{X} = \emptyset$
4 for each clique $\mathcal{M} \in \mathcal{C}$ do
5     $\mathcal{X} \leftarrow \mathcal{X} \cup \{$centroid($\mathcal{M}$)$\}$
6 return $\mathcal{X}$
```

Two vertices are connected with an edge if the corresponding prediction locations are within a distance $\sqrt{2}L$. An example of a constructed graph for a two dimensional problem is shown in Figure 6.4. The next step is to compute the clique centroids. Ideally, we would like to find maximum cliques in the graph. Unfortunately, finding maximum cliques is NP-Hard [29]. We limit ourselves to finding *maximal* cliques from each vertex in the graph since this can be done efficiently with a greedy algorithm: for each vertex $v \in V$ in the graph, grow the clique one vertex at a time by looping through the remaining vertices, add it to the clique if it is adjacent to every vertex in the clique and discard it otherwise (Line 2, maximalCliques). Note, this method does not yield *all* maximal cliques like the Bron-Kerbosch algorithm [15], which has an exponential time complexity in the worst case. Once we have the set of maximal cliques, the final step is to loop through the cliques and compute the centroid of the prediction locations associated with the clique (Line 5).

Centroid-Greedy

Proposition 3 ensures that the prediction locations are reasonable approximate solutions when the prediction locations are separated by a distance greater than $\sqrt{2}L$. Instead of Grid-Greedy which has a runtime of $O(\rho^d k^3 \max\{k, |\Omega|\})$ for Problem 1 (see Proposition 3), we remove the dependence on ρ^d i.e. the grid discretization, by limiting the search to the set of centroids (computed in Algorithm 8) and the set of prediction locations: $\mathcal{X} \cup \Omega$. The set of centroids is a feasible set for Problem 1 since $\Omega \subset \Theta$ and Θ is a convex set i.e. the set of measurement locations Θ contains the centroids of any subset of prediction locations. Since the number of maximal cliques computed by Algorithm 8 is bounded above by the number of prediction locations, the runtime of Centroid-Greedy is $O(|\Omega| k^3 \max\{k, |\Omega|\})$. This is an improvement over the runtime $O(\rho^d k^3 \max\{k, |\Omega|\})$ of Grid-Greedy, as long as $|\mathcal{X} \cup \Omega| < \rho^d$, which we will show in Section 6.5, is required for Grid-Greedy to obtain

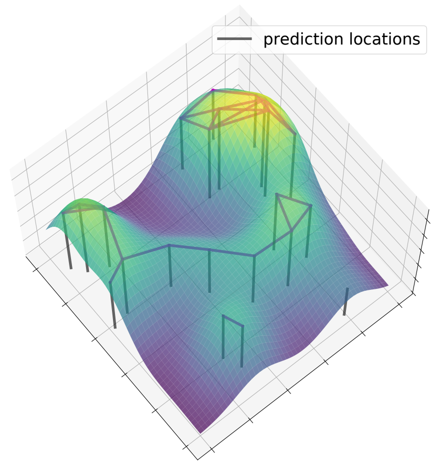


Figure 6.4: The objective function $f(S)$ when the budget $k = 1$ for a given set of prediction locations in two dimensions. Two prediction locations are connected by an edge if the distance between them is less than or equal to $\sqrt{2}L$.

good solutions for large fields. The steps for CENTROID-GREEDY are given in Algorithm 9.

```
Algorithm 9: CENTROID-GREEDY
   Input: Continuous Field: $\Theta$, Prediction Locations: $\Omega$, budget $k > 0$
   Output: Measurement Set: $S \subset \Theta$, $|S| = k$
1  $\mathcal{V} = $ MAXIMALCLIQUECENTROIDS($\Omega$)
2  Initialize $S_0 = \emptyset$
3  for $i = 1$ to $k$ do
4      $S_i = S_{i-1} \cup \{\arg\max_{x \in \mathcal{V}} f(S_{i-1} \cup \{x\}) - f(S_{i-1})\}$
5  return $S_k$
```

6.4.3 Implementation of the Greedy Algorithm

Each step of the greedy algorithm requires computing the maximizer of the marginal gain $f(S_i \cup \{x\}) - f(S_i)$ over all feasible x. Computing $f(S)$ in the form in (6.9) is time consuming and is not amenable to vectorization in NumPy [62] directly. Using Proposition

4, the marginal can be rewritten in a form that can be vectorized, and in practice is much faster to compute. For example, computing the solution for 500 prediction points, ground set size of 400, and a budget of 25 takes ≈ 0.5 seconds with vectorization and ≈ 14 seconds with the non-vectorized version.

Proposition 4. The marginal improvement of $f(S)$ when adding an element x to a set A is given by:

$$f(A \cup \{x\}) - f(A) = T_x \sum_{y \in \Omega} \left(R_{x,y}(A) - \phi_{\text{SE}}^2(x-y) \right)^2, \tag{6.29}$$

where $T_x = \left(\sigma_0^2 + \sigma^2 - \boldsymbol{b}_x^T(A)\, C(A)^{-1} \boldsymbol{b}_x(A) \right)^{-1}$ and $R_{x,y}(A) = \boldsymbol{b}_x(A)^T C(A)^{-1} \boldsymbol{b}_y(A)$.

Proof. We can partition the covariance matrix $C(A \cup \{x\})$ as follows:

$$C(A \cup \{x\}) = \begin{bmatrix} C(A) & \boldsymbol{b}_x(A) \\ \boldsymbol{b}_x(A)^T & \sigma_0^2 + \sigma^2 \end{bmatrix} \tag{6.30}$$

Define $T_x := \sigma_0^2 + \sigma^2 - \boldsymbol{b}_x(A)^T C(A)^{-1} \boldsymbol{b}_x(A)$. Using block matrix inversion,

$$C(A \cup \{x\}) = \begin{bmatrix} C(A)^{-1} + C(A)^{-1}\boldsymbol{b}_x(A) T_x \boldsymbol{b}_x(A)^T C(A)^{-1} & -C(A)^{-1}\boldsymbol{b}_x(A) T_x \\ -T_x \boldsymbol{b}_x(A)^T C(A)^{-1} & T_x \end{bmatrix} \tag{6.31}$$

Consider the objective

$$f(A \cup \{x\}) = \sum_{y \in \Omega} \boldsymbol{b}_y(A \cup \{x\}) C(A \cup \{x\})^{-1} \boldsymbol{b}_y(A \cup \{x\}). \tag{6.32}$$

We can partition $\boldsymbol{b}_y(A \cup \{x\})$ as follows:

$$\boldsymbol{b}_y(A \cup \{x\}) = \begin{bmatrix} \boldsymbol{b}_y(A) \\ \phi_{\text{SE}}(x-y) \end{bmatrix} \tag{6.33}$$

Define $R_{x,y}(A) = \boldsymbol{b}_x(A)^T C(A)^{-1} \boldsymbol{b}_y(A)$. Then, plugging in the required quantities in the objective and performing the vector-matrix multiplications results in

$$\begin{aligned} f(A \cup \{x\}) = f(A) + \sum_{y \in \Omega} \Big(& T_x R_{x,y}(A)^T R_{x,y}(A) \\ & - 2\phi_{\text{SE}}(x-y) T_x R_{x,y}(A)^T \\ & + \phi_{\text{SE}}^2(x-y) T_x \Big) \\ f(A \cup \{x\}) - f(A) = T_x \sum_{y \in \Omega} & \left(R_{x,y}(A) - \phi_{\text{SE}}(x-y) \right)^2. \end{aligned} \tag{6.34}$$

□

6.5 Numerical Results

We provide evidence of two advantages of Centroid-Greedy over Grid-Greedy. First, Centroid-Greedy obtains higher quality solutions on problem instances where the run time of both algorithms is comparable. Second, on instances where the solution quality is comparable, Centroid-Greedy finds the solution faster than Grid-Greedy. The solution quality is measured by the mean squared error (Equation 6.5) and the run time is measured in seconds.

Experimental Setup

We follow the setup in [122] where a Gaussian Process was fit to a real world dataset of organic matter measurements in an agricultural field [96]. Note that we do not require the actual data, only the parameters of the squared exponential covariance function and the variance of the measurement noise. Specifically, the authors [122] computed $L = 8.33$ meters, $\sigma_0 = 12.87$, and $\sigma^2 = 0.0361$. The interpretation of L is the distance one has to travel before the underlying function value changes [109]. Since the covariance function is a squared exponential, only the relative distances between points matter, not the absolute positions. This enables us to consider different environment sizes:

1. $D_{\text{small}} := \{(x, y) \in \mathbb{R}^2 : 0 \leq x \leq 40, 0 \leq y \leq 40\}$, Area $= 1600$ square meters.
2. $D_{\text{med}} := \{(x, y) \in \mathbb{R}^2 : 0 \leq x \leq 120, 0 \leq y \leq 120\}$, Area $= 14,400$ square meters.
3. $D_{\text{large}} := \{(x, y) \in \mathbb{R}^2 : 0 \leq x \leq 600, 0 \leq y \leq 600\}$, Area $= 360,000$ square meters.

We also consider three regimes for the number of prediction points: sparse (20 points, budget 8), moderate (300 points, budget 75), and dense (1000 points, budget 200). The results in the following sections are based on 10 randomly generated problem instances for each combination of environment type (small, medium, large) and prediction point regime (sparse, moderate, dense). The experiments are implemented using NumPy [62] on an AMD Ryzen 7 2700 processor.

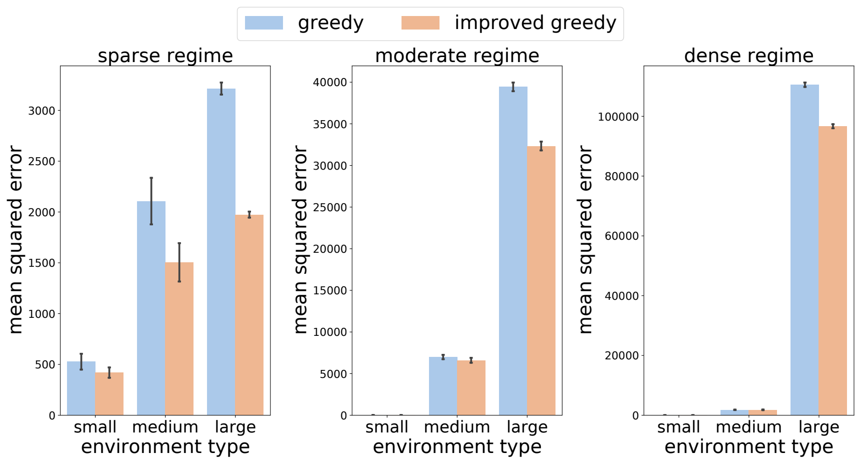


Figure 6.5: Comparison of the solution quality while keeping the run time approximately the same. Centroid-Greedy obtains equal or better solutions Grid-Greedy in all environment types and regime of prediction points.

6.5.1 Solution Quality

In the first set of experiments, we aim to answer the following question: given the same computational resources, which algorithm provides a better solution? To ensure equal computational resources, for a $N \times N$ grid discretization, we set $N = \left\lceil \sqrt{2|\Omega|} \right\rceil$. Since the number of maximal cliques computed in Algorithm 8 is at most $|\Omega|$, this ensures the runtimes are comparable. The grids selected are: 7×7 (sparse regime), 25×25 (moderate regime), 45×45 (dense regime).

The results are shown in Figure 6.5. In the sparse regime (left plot) Centroid-Greedy outperforms Grid-Greedy on average in all environment types. The difference in performance is the highest in large environments since the grid resolution is not sufficient to cover the space. In the moderate regime (center plot) and dense regime (right plot), the solution quality of both algorithms is similar in small and medium sized environments. However, for large environments, Centroid-Greedy obtains better solutions. The difference in performance reduces as we move from the sparse to dense regime. This is because the high density of prediction points increases the chance of close proximity with grid points, even in the case of low resolution grids. In summary, using a similar amount of computational resources, Centroid-Greedy obtains solutions of equal or higher quality than Grid-Greedy across all environment sizes and regimes on the number of prediction points.

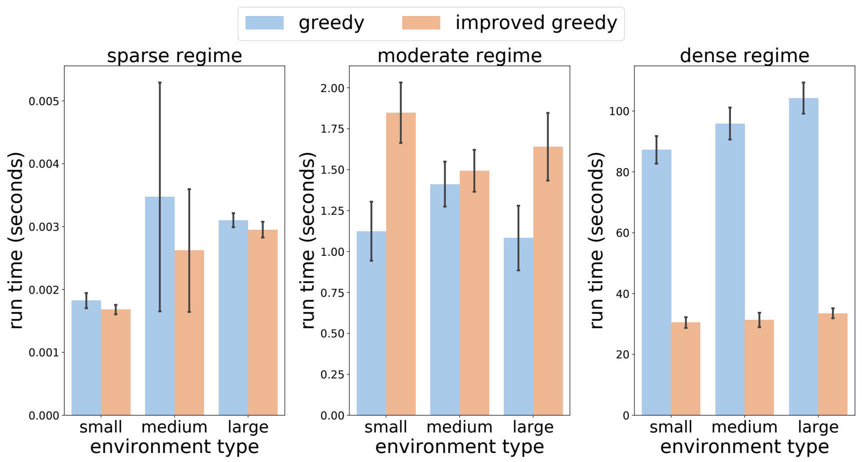


Figure 6.6: Comparison of the run time while keeping the solution quality approximately the same. GRID-GREEDY practically takes at least as much time as CENTROID-GREEDY to find solutions of similar quality.

6.5.2 Run Time

In the second set of experiments, we aim to answer the following question: how much longer does it take for GRID-GREEDY to achieve similar solution quality as CENTROID-GREEDY? For each problem instance, if CENTROID-GREEDY obtains a higher objective value than GRID-GREEDY we repeatedly increase the grid resolution until GRID-GREEDY attains the a similar objective value. We compare the time taken by GRID-GREEDY on the final grid resolution to the time taken by CENTROID-GREEDY.

The results are shown in Figure 6.6. In the sparse regime (left plot) and moderate regime (center plot), the run times are practically the same. In the moderate regime, the number of prediction points is a bit higher than the number of grid points which is the reason for the slightly higher runtime of CENTROID-GREEDY. However, in the dense regime, GRID-GREEDY takes approximately 2.5 times (small environments), 4 times (medium size environments), and 5 times (large environments) as long as CENTROID-GREEDY to attain a similar objective value. The runtime of GRID-GREEDY increases with the size of the environment, while the runtime of CENTROID-GREEDY remains fairly constant. Note, while the run times in our experiments seem feasible in practice, the fields considered in these experiments are small compared to average farm sizes. For example, in 2019, the average farm size in USA was 444 acres [129] which is 5 times the size of the largest environment considered here. We expect larger reductions in run time for these agricultural fields in practice. In summary, Centroid-Greedy finds solutions of similar quality to Grid-Greedy more efficiently i.e. using less or equal amounts of time, across all environments and regimes on the number of prediction points.

6.6 Summary

We studied the problem of selecting the k-best measurements to estimate a spatial field at a finite set of locations. We proposed an algorithm based on computing maximal cliques that outperformed a grid discretization of the field, both in terms of solution quality and runtime.

Chapter 7

Conclusions

Linear estimation under resource constraints is a recurring challenge in many domains. In this thesis, we proposed algorithms to solve it under a variety of interesting settings including sensor scheduling, informative path planning, and spatial sampling. Since the general problem is NP-hard, one does not expect an algorithm to simultaneously a) compute optimal solutions b) in polynomial time c) for any instance. In Chapters 3 and 4, we relax the requirement of polynomial-time solvability and developed mixed integer programs to compute optimal solutions to sensor scheduling and informative path planning. In Chapters 5 and 6, we relax the requirement of finding optimal solutions and develop approximation algorithms for minimum resource sampling and tour planning.

7.1 Summary

In Chapter 3, we studied the generalized version of the sensor scheduling problem capturing problems such as sensor placement, scheduling, and LQG sensing design. Our approach was rooted in mixed integer optimization. We formulated a mixed integer quadratic program by exploiting the optimality of the Kalman filter. In simulations, we showed the effectiveness of the approach in computing optimal solutions to systems with 30 to 50 states. In addition, the solver also returned better quality solutions over the popular greedy algorithm when constrained to time out within a few seconds. Looking forward, it would be interesting to devise a more compact formulation as the number of decision variables in the proposed approach grows quadratically with the time horizon. This leads to large running times for problems with long horizons.

In Chapter 4, we proposed the first computationally tractable MIP for IPP in GPs. Our approach was rooted in exploiting the optimality of the GP posterior mean for matrix non-decreasing functions of the expected squared estimation error which enabled the formulation of the resulting MIP. Using standard network flow techniques, our approach was also able to handle typical routing constraints that arise in IPP. While the runtimes of MIP are generally concerning, we showed in simulation that the proposed MIPs can be solved to optimality in several settings (high connectivity, large test sets, multiple robots, large budgets) and when terminated early, return solutions of significantly higher quality than the commonly used greedy algorithm. Our work is a first step towards exact algorithms, an unexplored avenue in IPP. Looking forward, the major question is how to use this optimal offline planner in the case when the model is not well-specified. The typical approach is to plan in a receding horizon fashion while collecting data, updating the model, and using the offline planner to generate feasible paths. It would be interesting to study the empirical performance of the optimal offline planner in this setting.

In Chapter 5, we considered the problem of finding the subset of sample locations and the shortest tour in a spatial field that guarantees a desired level of uncertainty via the posterior variance in a Gaussian Process. We provided approximation algorithms for both problems in two settings: convex environments and finite test sets. The central idea for all algorithms was to compute minimize size covers of the test set in a way that guarantees feasibility for the problems at hand. In convex environments, the algorithms computed hexagonal covers while for finite test sets, the algorithms relied on approximately solving the set cover problem. We also provided a counterexample to disprove a claim on a lower bound for Sample Placement. Looking forward, it would be interesting to characterize the sub-optimality of the proposed algorithms when the model is misspecified, an assumption that is often violated in practice.

In Chapter 6, we discussed the problem of selecting a k-subset that yields the best linear estimate at a set of prediction locations in a continuous spatial field. One approach is to solve the problem using a grid discretization of the field and greedily select k measurement locations. However, this can be computationally expensive for large fields. Instead, we restricted the search of the greedy algorithm to the set of prediction locations and the centroids of their cliques. This was motivated by identifying a critical distance between two prediction points which characterized the optimal solution in 1D. In simulations, we showed the effectiveness of the proposed approach in terms of solution quality and runtime.

7.2 Future Work

While our work has answered a few questions on the front of linear estimation with resource constraints, we believe there are many challenges yet to be solved. The two critical ones we highlight for future work are model uncertainty and stronger exact formulations.

Model Uncertainty One of the limitations of our work is that we have assumed the models are fully specified. For example, in Chapter 3 we assumed full knowledge of the system dynamics and measurement model. In Chapters 4 and 5, we assumed perfect knowledge of the GP kernel hyperparameters. This is a strong assumption that may be violated in practice since the model is learnt from the collected data. While one can use offline planners in-the-loop (plan samples/paths, execute partial solution, collect data, update model, and replan), the guarantees of optimality on the error are now lost. One alternative is an adaptive algorithm where the decisions depends on the data collected thus far. One direction that would be interesting is to characterize the benefit of adaptivity over the optimal offline algorithm.

Exact Formulations The strength of MIPs is highly dependent on their formulation. In Chapter 4, we used the SOS Type-1 constraint to model the relationship between a visited vertex and its associated coefficient in the linear estimator. These are frequently recast as big-M constraints which are known to yield weak formulations leading to long solve times in practice. There are other ways to model this relationship, most notably the perspective reformulation [57] which has been exploited for high dimensional sparse regression [65]. Given the close connection of IPP and sensor scheduling to sparse regression, we believe this would be an interesting direction to pursue which could potentially yield faster runtimes. Further, the relaxations of the MIPs could be used to generate approximate solutions quickly via convex optimization. For the case of sensor scheduling, we believe a minimal formulation that solely contains integer variables would be desirable. This is because the number of continuous variables in the proposed MIQP scales quadratically with the time horizon whereas the number of integer variables only depends on the number of sensors.

References

[1] Pankaj K Agarwal and Jiangwei Pan. Near-Linear Algorithms for Geometric Hitting Sets and Set Covers. In *Proceedings of the Annual Symposium on Computational Geometry*, 2014.

[2] Brian DO Anderson and John B Moore. *Optimal Filtering.* Courier Corporation, 2012.

[3] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Decentralized Active Information Acquisition: Theory and Application to Multi-Robot SLAM. In *International Conference on Robotics and Automation (ICRA)*, 2015.

[4] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 1998.

[5] D. Bertsimas, A. King, and R. Mazumder. Best Subset Selection via a Modern Optimization Lens. *The Annals of Statistics*, 2016.

[6] Dimitris Bertsimas and Bart Van Parys. Sparse High-Dimensional Regression: Exact Scalable Algorithms and Phase Transitions. *The Annals of Statistics*, 48(1):300 – 323, 2020.

[7] Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. Sparse regression: Scalable algorithms and empirical performance. *Statistical Science*, 2020.

[8] Dimitris Bertsimas and Bart Van Parys. Sparse High-Dimensional Regression: Exact Scalable Algorithms and Phase Transitions. *The Annals of Statistics*, 2020.

[9] Dimitris Bertsimas and Robert Weismantel. *Optimization Over Integers.* Dynamic Ideas Belmont, 2005.

[10] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Informative Path Planning for an Autonomous Underwater Vehicle. In *International Conference on Robotics and Automation (ICRA)*, 2010.

[11] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Optimizing Waypoints for Monitoring Spatiotemporal Phenomena. *The International Journal of Robotics Research*, 2013.

[12] Jonathan Binney and Gaurav S Sukhatme. Branch and Bound for Informative Path Planning. In *Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[13] Avrim Blum, Shuchi Chawla, David R Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *SIAM Journal on Computing*, 2007.

[14] Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[15] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[16] Xiaoyi Cai, Brent Schlotfeldt, Kasra Khosoussi, Nikolay Atanasov, George J Pappas, and Jonathan P How. Energy-Aware, Collision-Free Information Gathering for Heterogeneous Robot Teams. *IEEE Transactions on Robotics*, 2023.

[17] Christopher I Calle and Shaunak D Bopardikar. Probabilistic Performance Bounds for Randomized Sensor Selection in Kalman Filtering. In *American Control Conference (ACC)*, pages 4395–4400. IEEE, 2021.

[18] Nannan Cao, Kian Hsiang Low, and John M Dolan. Multi-Robot Informative Path Planning for Active Sensing of Environmental Phenomena: a Tale of Two Algorithms. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 7–14, 2013.

[19] Avishy Carmi and Pini Gurfil. Sensor Selection via Compressed Sensing. *Automatica*, 49(11):3304–3314, 2013.

[20] Luiz FO Chamon, George J Pappas, and Alejandro Ribeiro. Approximate Supermodularity of Kalman Filter Sensor Selection. *IEEE Transactions on Automatic Control*, 2020.

[21] Hai-Chau Chang and Lih-Chung Wang. A Simple Proof of Thue's Theorem on Circle Packing. *arXiv preprint arXiv:1009.4322*, 2010.

[22] I-Ming Chao, Bruce L Golden, and Edward A Wasil. The Team Orienteering Problem. *European Journal of Operational Research*, 1996.

[23] Chandra Chekuri, Nitish Korula, and Martin Pál. Improved Algorithms for Orienteering and Related Problems. *Transactions on Algorithms (TALG)*, 2012.

[24] Chandra Chekuri and Martin Pal. A Recursive Greedy Algorithm for Walks in Directed Graphs. In *Symp. on Found. of Comp. Sci. (FOCS)*, 2005.

[25] Weizhe Chen, Roni Khardon, and Lantao Liu. Adaptive Robotic Information Gathering via Non-Stationary Gaussian Processes. *arXiv preprint arXiv:2306.01263*, 2023.

[26] Dean T Connor et al. Radiological Mapping of Post-Disaster Nuclear Environments using Fixed-Wing Unmanned Aerial Systems: a Study from Chornobyl. *Frontiers in Robotics and AI*, 2020.

[27] William J Cook, William H Cunningham, William R Pulleyblank, and Alexander Schrijver. *Combinatorial Optimisation.* Springer, 1998.

[28] Micah Corah and Nathan Michael. Distributed Matroid-Constrained Submodular Maximization for Multi-Robot Exploration: Theory and Practice. *Autonomous Robots*, 2019.

[29] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms.* MIT press, 2009.

[30] Jorge Cortes. Distributed Kriged Kalman Filter for Spatial Estimation. *IEEE Transactions on Automatic Control*, 2009.

[31] Jorge Cortés. Distributed kriged kalman filter for spatial estimation. *IEEE Trans. on Automatic Control*, 54:2816–2827, 2009.

[32] IBM ILOG Cplex. V12. 1: User's Manual for CPLEX. *International Business Machines Corporation*, 2009.

[33] Noel Cressie. The origins of kriging. *Mathematical geology*, 22:239–252, 1990.

[34] Noel Cressie. *Statistics for Spatial Data.* John Wiley & Sons, 2015.

[35] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America*, 1954.

[36] Abhimanyu Das and David Kempe. Algorithms for Subset Selection in Linear Regression. In *ACM Symposium on Theory of Computing*, 2008.

[37] Abhimanyu Das and David Kempe. Approximate Submodularity and its Applications: Subset Selection, Sparse Approximation and Dictionary Selection. *The Journal of Machine Learning Research*, 2018.

[38] Jnaneshwar Das, Frédéric Py, Julio BJ Harvey, John P Ryan, Alyssa Gellene, Rishi Graham, David A Caron, Kanna Rajan, and Gaurav S Sukhatme. Data-driven Robotic Sampling for Marine Ecosystem Monitoring. *The International Journal of Robotics Research*, 2015.

[39] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms*. McGraw-Hill Higher Education New York, 2008.

[40] Neil K Dhingra, Mihailo R Jovanović, and Zhi-Quan Luo. An ADMM Algorithm for Optimal Sensor and Actuator Selection. In *Conference on Decision and Control*, pages 4039–4044. IEEE, 2014.

[41] Adrian Dumitrescu and Joseph SB Mitchell. Approximation Algorithms for TSP with Neighborhoods in the Plane. *Journal of Algorithms*, 2003.

[42] Adrian Dumitrescu and Csaba D Tóth. Constant-Factor Approximation for TSP with Disks. *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, 2017.

[43] Marco A Duran and Ignacio E Grossmann. An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs. *Mathematical Programming*, 36:307–339, 1986.

[44] Shamak Dutta, Nils Wilde, and Stephen L. Smith. An Improved Greedy Algorithm for Subset Selection in Linear Estimation. In *2022 European Control Conference (ECC)*, pages 1067–1072, 2022.

[45] Shamak Dutta, Nils Wilde, and Stephen L Smith. Informative Path Planning in Random Fields via Mixed Integer Programming. In *61st Conference on Decision and Control (CDC)*, 2022.

[46] Shamak Dutta, Nils Wilde, and Stephen L Smith. A Unified Approach to Optimally Solving Sensor Scheduling and Sensor Selection Problems in Kalman Filtering. *Conference on Decision and Control (CDC)*, 2023. To appear.

[47] Shamak Dutta, Nils Wilde, Pratap Tokekar, and Stephen L. Smith. Approximation Algorithms for Robot Tours in Random Fields with Guaranteed Estimation Accuracy. In *Int. Conf. on Robotics and Automation (ICRA)*, 2023.

[48] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. Log-det Heuristic for Matrix Rank Minimization with Applications to Hankel and Euclidean Distance Matrices. In *American Control Conference*, 2003.

[49] Uriel Feige. A Threshold of ln n for Approximating Set Cover. *Journal of the ACM (JACM)*, 1998.

[50] L Fejes. Über die dichteste kugellagerung. *Mathematische Zeitschrift*, 48(1):676–684, 1942.

[51] Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal Packing and Covering in the Plane are NP-complete. *Information processing letters*, 1981.

[52] Zachary Friggstad, Sreenivas Gollapudi, Kostas Kollias, Tamas Sarlos, Chaitanya Swamy, and Andrew Tomkins. Orienteering Algorithms for Generating Travel Itineraries. In *International Conference on Web Search and Data Mining*, 2018.

[53] Zachary Friggstad and Chaitanya Swamy. Compact, Provably-Good LPs for Orienteering and Regret-Bounded Vehicle Routing. In *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 2017.

[54] Bruce L Golden, Larry Levy, and Rakesh Vohra. The Orienteering Problem. *Naval Research Logistics*, 1987.

[55] Mohinder S Grewal and Angus P Andrews. Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.

[56] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering Problem: A Survey of Recent Variants, Solution Approaches and Applications. *European Journal of Operational Research*, 2016.

[57] Oktay Günlük and Jeff Linderoth. Perspective Reformulations of Mixed Integer Nonlinear Programs with Indicator Variables. *Mathematical Programming*, 2010.

[58] Oktay Günlük and Jeff Linderoth. Perspective Reformulation and Applications. In *Mixed Integer Nonlinear Programming*, pages 61–89. Springer, 2011.

[59] Vijay Gupta, Timothy H Chung, Babak Hassibi, and Richard M Murray. On a Stochastic Sensor Selection Algorithm with Applications in Sensor Scheduling and Sensor Coverage. *Automatica*, 42(2):251–260, 2006.

[60] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[61] Isabelle Guyon et al. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[62] Charles R. Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[63] Abolfazl Hashemi, Mahsa Ghasemi, Haris Vikalo, and Ufuk Topcu. Randomized Greedy Sensor Selection: Leveraging Weak Submodularity. *IEEE Transactions on Automatic Control*, 66(1):199–212, 2020.

[64] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York Inc., 2001.

[65] Hussein Hazimeh, Rahul Mazumder, and Ali Saab. Sparse Regression at Scale: Branch-and-Bound Rooted in First-Order Optimization. *Mathematical Programming*, 2022.

[66] Keld Helsgaun. An Effective Implementation of the Lin–Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research*, 2000.

[67] Gregory Hitz, Enric Galceran, Marie-Ève Garneau, François Pomerleau, and Roland Siegwart. Adaptive Continuous-Space Informative Path Planning for Online Environmental Monitoring. *Journal of Field Robotics*, 2017.

[68] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based Robotic Information Gathering Algorithms. *The International Journal of Robotics Research*, 2014.

[69] Syed Talha Jawaid and Stephen L Smith. Submodularity and Greedy Algorithms in Sensor Scheduling for Linear Dynamical Systems. *Automatica*, 61:282–288, 2015.

[70] Donald B Johnson. Finding all the Elementary Circuits of a Directed Graph. *SIAM Journal on Computing*, 1975.

[71] Siddharth Joshi and Stephen Boyd. Sensor Selection via Convex Optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2008.

[72] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear Estimation*. Prentice Hall, 2000.

[73] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[74] Frank J Kelly and Julia Kelly. London air quality: A real world experiment in progress. *Biomarkers*, 14:5–11, 2009.

[75] Akira Kohara, Kunihisa Okano, Kentaro Hirata, and Yukinori Nakamura. Sensor Placement Minimizing the State Estimation Mean Square Error: Performance Guarantees of Greedy Solutions. In *Conference on Decision and Control*, pages 1706–1711. IEEE, 2020.

[76] Bernhard H Korte and Jens Vygen. *Combinatorial Optimization*. Springer, 2011.

[77] Andreas Krause, H Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust Submodular Observation Selection. *Journal of Machine Learning Research*, 9(12), 2008.

[78] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient algorithms and Empirical Studies. *Journal of Machine Learning Research*, 2008.

[79] Xiaodong Lan and Mac Schwager. Rapidly Exploring Random Cycles: Persistent Estimation of Spatiotemporal Fields with Multiple Sensing Robots. *IEEE Transactions on Robotics*, 2016.

[80] Sören Laue, Matthias Mitterreiter, and Joachim Giesen. Computing Higher Order Derivatives of Matrix and Tensor Expressions. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[81] Sören Laue, Matthias Mitterreiter, and Joachim Giesen. A Simple and Efficient Tensor Calculus. In *AAAI Conference on Artificial Intelligence, (AAAI)*. 2020.

[82] Jerome Le Ny and George J Pappas. On Trajectory Optimization for Active Sensing in Gaussian Process Models. In *Conference on Decision and Control (CDC)*, pages 6286–6292. IEEE, 2009.

[83] Kai-Chieh Ma, Lantao Liu, Hordur K Heidarsson, and Gaurav S Sukhatme. Data-driven Learning and Planning for Environmental Sampling. *Journal of Field Robotics*, 2018.

[84] Dipankar Maity, David Hartman, and John S Baras. Sensor Scheduling for Linear Systems: A Covariance Tracking Approach. *Automatica*, 136:110078, 2022.

[85] Roman Marchant and Fabio Ramos. Bayesian Optimisation for Intelligent Environmental Monitoring. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[86] Tobia Marcucci, Jack Umenberger, Pablo A Parrilo, and Russ Tedrake. Shortest Paths in Graphs of Convex Sets. *arXiv preprint arXiv:2101.11565*, 2021.

[87] Seth McCammon and Geoffrey A Hollinger. Topological Hotspot Identification for Informative Path Planning with a Marine Robot. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[88] Seth McCammon, Gilberto Marcon dos Santos, Matthew Frantz, Timothy P Welch, Graeme Best, R Kipp Shearman, Jonathan D Nash, John A Barth, Julie A Adams, and Geoffrey A Hollinger. Ocean Front Detection and Tracking using a Team of Heterogeneous Marine Vehicles. *Journal of Field Robotics*, 2021.

[89] Ajith Anil Meera, Marija Popović, Alexander Millane, and Roland Siegwart. Obstacle-aware Adaptive Informative Path Planning for Uav-based Target Search. In *International Conference on Robotics and Automation (ICRA)*, 2019.

[90] Alexandra Meliou, Andreas Krause, Carlos Guestrin, and Joseph M Hellerstein. Nonmyopic Informative Path Planning in Spatio-temporal Models. In *AAAI*, 2007.

[91] P Miliotis. Using Cutting Planes to Solve the Symmetric Travelling Salesman Problem. *Mathematical Programming*, 1978.

[92] Alan Miller. *Subset Selection in Regression.* CRC Press, 2002.

[93] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM (JACM)*, 1960.

[94] Yilin Mo, Roberto Ambrosino, and Bruno Sinopoli. Sensor Selection Strategies for State Estimation in Energy Constrained Wireless Sensor Networks. *Automatica*, 47(7):1330–1338, 2011.

[95] Nima Moshtagh, Lingji Chen, and Raman Mehra. Optimal Measurement Selection for Any-Time Kalman Filtering with Processing Constraints. In *Conference on Decision and Control*, pages 5074–5079. IEEE, 2009.

[96] DJ Mulla, AC Sekely, and M Beatty. Evaluation of remote sensing and targeted soil sampling for variable rate application of nitrogen. In *International Conference on Precision Agriculture*, pages 1–15, Bloomington, MN, USA, July 2000.

[97] Ulrich Münz, Maximilian Pfister, and Philipp Wolfrum. Sensor and Actuator Placement for Linear Systems Based on H_2 and H_∞ Optimization. *IEEE Transactions on Automatic Control*, pages 2984–2989, 2014.

[98] Patrenahalli M. Narendra and Keinosuke Fukunaga. A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on computers*, 26(09):917–922, 1977.

[99] Balas Kausik Natarajan. Sparse Approximate Solutions to Linear Systems. *SIAM Journal on Computing*, 1995.

[100] George L Nemhauser et al. An Analysis of Approximations for Maximizing Submodular Set Functions—I. *Mathematical programming*, 1978.

[101] Yurii Nesterov. *Lectures on Convex Optimization*. Springer, 2018.

[102] Alex Olshevsky. Minimal Controllability Problems. *IEEE Transactions on Control of Network Systems*, 1(3):249–258, 2014.

[103] Gábor Pataki. Teaching Integer Programming Formulations using the Traveling Salesman Problem. *SIAM review*, 2003.

[104] Julio A Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A Castellanos. A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers. *IEEE Transactions on Robotics*, 2023.

[105] Marija Popović, Teresa Vidal-Calleja, Jen Jen Chung, Juan Nieto, and Roland Siegwart. Informative Path Planning for Active Field Mapping under Localization Uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[106] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An Informative Path Planning Framework for UAV-based Terrain Monitoring. *Autonomous Robots*, 2020.

[107] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Inkyu Sa, Roland Siegwart, and Juan Nieto. Multiresolution Mapping and Informative path Planning for Uav-based Terrain Monitoring. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[108] Naren Ramakrishnan et al. Gaussian Processes for Active Data Mining of Spatial Aggregates. In *SIAM International Conference on Data Mining*, 2005.

[109] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[110] Julius Rückin, Liren Jin, and Marija Popović. Adaptive Informative Path Planning using Deep Reinforcement Learning for UAV-based Active Sensing. In *International Conference on Robotics and Automation (ICRA)*, 2022.

[111] Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J Pappas. Anytime Planning for Decentralized Multirobot Active Information Gathering. *IEEE Robotics and Automation Letters*, 2018.

[112] Shamaiah, Manohar and Banerjee, Siddhartha and Vikalo, Haris. Greedy Sensor Selection: Leveraging Submodularity. In *Conference on Decision and Control (CDC)*, pages 2572–2577. IEEE, 2010.

[113] Milad Siami and Ali Jadbabaie. A Separation Theorem for Joint Sensor and Actuator Scheduling with Guaranteed Performance Bounds. *Automatica*, 119:109054, 2020.

[114] Milad Siami, Alexander Olshevsky, and Ali Jadbabaie. Deterministic and Randomized Actuator Scheduling with Guaranteed Performance Bounds. *IEEE Transactions on Automatic Control*, 66(4):1686–1701, 2020.

[115] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J Kaiser. Efficient Informative Sensing using Multiple Robots. *Journal of Artificial Intelligence Research*, 2009.

[116] Prince Singh, Min Chen, Luca Carlone, Sertac Karaman, Emilio Frazzoli, and David Hsu. Supermodular Mean Squared Error Minimization for Sensor Scheduling in Optimal Kalman Filtering. In *American Control Conference (ACC)*, pages 5787–5794. IEEE, 2017.

[117] Ryan N Smith, Mac Schwager, Stephen L Smith, Burton H Jones, Daniela Rus, and Gaurav S Sukhatme. Persistent Ocean Monitoring with Underwater Gliders: Adapting Sampling Resolution. *Journal of Field Robotics*, 2011.

[118] Petr Somol, Pavel Pudil, and Josef Kittler. Fast branch & bound algorithms for optimal feature selection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(7):900–912, 2004.

[119] Paul Stankiewicz, Yew T Tan, and Marin Kobilarov. Adaptive Sampling with an Autonomous Underwater Vehicle in Static Marine Environments. *Journal of Field Robotics*, 2021.

[120] Tyler H Summers, Fabrizio L Cortesi, and John Lygeros. On Submodularity and Controllability in Complex Dynamical Networks. *IEEE Transactions on Control of Network Systems*, 3(1):91–101, 2015.

[121] Varun Suryan and Pratap Tokekar. Learning a Spatial Field with Gaussian Process Regression in Minimum Time. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 301–317. Springer, 2018.

[122] Varun Suryan and Pratap Tokekar. Learning a Spatial Field in Minimum Time with a Team of Robots. *IEEE Transactions on Robotics*, 2020.

[123] Onur Tekdas, Deepak Bhadauria, and Volkan Isler. Efficient Data Collection from Wireless Nodes under the Two-ring Communication Model. *The International Journal of Robotics Research*, 31(6):774–784, 2012.

[124] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. *IEEE Transactions on Robotics*, 2016.

[125] Vasileios Tzoumas, Luca Carlone, George J Pappas, and Ali Jadbabaie. LQG Control and Sensing Co-Design. *IEEE Transactions on Automatic Control*, 66(4):1468–1483, 2020.

[126] Vasileios Tzoumas, Ali Jadbabaie, and George J Pappas. Near-optimal Sensor Scheduling for Batch State Estimation: Complexity, Algorithms, and Limits. In *Conference on Decision and Control (CDC)*, pages 2695–2702. IEEE, 2016.

[127] Vasileios Tzoumas, Mohammad Amin Rahimian, George J Pappas, and Ali Jadbabaie. Minimal Actuator Placement with Bounds on Control Effort. *IEEE Transactions on Control of Network Systems*, pages 67–78, 2015.

[128] Tzoumas, Vasileios and Jadbabaie, Ali and Pappas, George J. Sensor Placement for Optimal Kalman Filtering: Fundamental Limits, Submodularity, and Algorithms. In *American Control Conference (ACC)*, 2016.

[129] USDA, National Agricultural Statistics Service. Farms and land in farms, 2019 summary, February 2020.

[130] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian Process Modeling of Large-Scale Terrain. *Journal of Field Robotics*, 2009.

[131] Michael P Vitus, Wei Zhang, Alessandro Abate, Jianghai Hu, and Claire J Tomlin. On Efficient Sensor Scheduling for Linear Dynamical Systems. *Automatica*, 48(10):2482–2493, 2012.

[132] Yin Wang, Mario Sznaier, and Fabrizio Dabbene. A Convex Optimization Approach to Worst-Case Optimal Sensor Selection. In *Conference on Decision and Control*, pages 6353–6358. IEEE, 2013.

[133] Richard Webster and Margaret A Oliver. *Geostatistics for environmental scientists*. John Wiley & Sons, 2007.

[134] James E Weimer, Bruno Sinopoli, and Bruce H Krogh. A Relaxation Approach to Dynamic Sensor Selection in Large-Scale Wireless Networks. In *International Conference on Distributed Computing Systems Workshops*, pages 501–506. IEEE, 2008.

[135] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press Cambridge, MA, 2006.

[136] Yihong Wu. Lecture Notes on ECE 598: Information-theoretic Methods in High-Dimensional Statistics. `http://www.stat.yale.edu/~yw562/teaching/it-stats.pdf`, 2016.

[137] Shiyi Yang, Nan Wei, Soo Jeon, Ricardo Bencatel, and Anouck Girard. Real-time optimal path planning and wind estimation using gaussian process regression for precision airdrop. In *American Control Conference (ACC)*, pages 2582–2587, 2017.

[138] Lintao Ye, Sandip Roy, and Shreyas Sundaram. On the Complexity and Approximability of Optimal Sensor Selection for Kalman Filtering. In *2018 Annual American Control Conference (ACC)*, pages 5049–5054. IEEE, 2018.

[139] Lintao Ye, Nathaniel Woodford, Sandip Roy, and Shreyas Sundaram. On the Complexity and Approximability of Optimal Sensor Selection and Attack for Kalman Filtering. *IEEE Transactions on Automatic Control*, 66(5):2146–2161, 2020.

[140] Jingjin Yu, Mac Schwager, and Daniela Rus. Correlated Orienteering Problem and its Application to Persistent Monitoring Tasks. *IEEE Transactions on Robotics*, 2016.

[141] Gioele Zardini, Andrea Censi, and Emilio Frazzoli. Co-design of Autonomous Systems: From Hardware Selection to Control synthesis. In *European Control Conference (ECC)*, pages 682–689. IEEE, 2021.

[142] Haifeng Zhang and Yevgeniy Vorobeychik. Submodular Optimization with Routing Constraints. In *AAAI Conference on Artificial Intelligence*, 2016.

[143] Haotian Zhang, Raid Ayoub, and Shreyas Sundaram. Sensor Selection for Kalman Filtering of Linear Dynamical Systems: Complexity, Limitations and Greedy Algorithms. *Automatica*, 78:202–210, 2017.

[144] Lin Zhao, Wei Zhang, Jianghai Hu, Alessandro Abate, and Claire J Tomlin. On the Optimal Solutions of the Infinite-Horizon Linear Sensor Scheduling Problem. *IEEE Transactions on Automatic Control*, 59(10):2825–2830, 2014.

[145] Hai Zhu, Jen Jen Chung, Nicholas RJ Lawrance, Roland Siegwart, and Javier Alonso-Mora. Online Informative Path Planning for Active Information Gathering of a 3d Surface. In *International Conference on Robotics and Automation (ICRA)*, 2021.