NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SHALLOW-WATER SENSOR PLACEMENT

by

Erik V. Vargas

September 2022

Thesis Advisor:	Robert L. Bassett
Co-Advisor:	Jefferson Huang
Second Reader:	Ross J. Schuchard

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
SHALLOW-WATER SENSOR PLACEMENT

5. FUNDING NUMBERS

6. AUTHOR(S) Erik V. Vargas

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Placing acoustic sensors allows for remote detection of vessels in areas of interest. We study the problem of placing only a few sensors to effectively monitor signals at many locations. Though our approach applies to sensor placement problems generally, we focus on placing hydrophones to efficiently monitor acoustic signals over a large region. Our starting point is the mutual information criterion for sensor placement, which despite being theoretically attractive has not been widely adopted because of its computational difficulty. To remedy these computational challenges, we introduce a novel branch and bound algorithm that relies upon a new semidefinite programming relaxation of the mutual information problem. Our contributions allow practitioners to solve sensor placement problems to global optimality while exploring only a fraction of the solution space required by brute force. Our work has been made open source at https://github.com/rbassett3/mutual_info_sensor_placement.

14. SUBJECT TERMS sensor placement, semidefinite program, mutual information, Goemans-Williamson	**15. NUMBER OF PAGES** 57
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

SHALLOW-WATER SENSOR PLACEMENT

Erik V. Vargas
Lieutenant, United States Navy
BS, University of California, Santa Barbara, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2022

Approved by: Robert L. Bassett
Advisor

Jefferson Huang
Co-Advisor

Ross J. Schuchard
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Placing acoustic sensors allows for remote detection of vessels in areas of interest. We study the problem of placing only a few sensors to effectively monitor signals at many locations. Though our approach applies to sensor placement problems generally, we focus on placing hydrophones to efficiently monitor acoustic signals over a large region. Our starting point is the mutual information criterion for sensor placement, which despite being theoretically attractive has not been widely adopted because of its computational difficulty. To remedy these computational challenges, we introduce a novel branch and bound algorithm that relies upon a new semidefinite programming relaxation of the mutual information problem. Our contributions allow practitioners to solve sensor placement problems to global optimality while exploring only a fraction of the solution space required by brute force. Our work has been made open source at https://github.com/rbassett3/mutual_info_sensor_placement.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Introduction 2
1.3 Outline 3

2 Mutual Information for Sensor Placement **5**
2.1 Mutual Information 5
2.2 Semidefinite Relaxations of Combinatorial Optimization Problems 7
2.3 SDP Relaxation of Maximizing Mutual Information 8

3 Branch and Bound Algorithm **11**
3.1 Partially Determined Subproblem 11
3.2 Upper and Lower Bounds 12
3.3 Implementation in Low-Level Languages 13

4 Computational Experiments **15**
4.1 Onion Method 15
4.2 Experiments 16
4.3 Application: Hydrophone Placement. 27
4.4 Conclusion 31

Appendix: Proofs and Definitions **33**

List of References **35**

Initial Distribution List **37**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Figure 4.1 Comparison of Branching Methods with Full Depth and No Rounding 20

Figure 4.2 Difference between Rounding and Not Rounding 21

Figure 4.3 Difference between Full Depth and Low Depth 24

Figure 4.4 Sensor Placements for Monitoring Vessel Traffic in the Monterey Bay 28

Figure 4.5 Acoustic Simulation Ray Tracing 30

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 4.1 Full-Depth without Rounding: Problem Size vs. SDPs Solved by Branching Strategy 18

Table 4.2 Full-Depth without Rounding: Problem Size vs. Leaf Nodes Explored by Branching Strategy 19

Table 4.3 Full-Depth with Rounding: Problem Size vs. SDPs Solved by Branching Strategy 22

Table 4.4 Full-Depth with Rounding: Problem Size vs. Leaf Nodes Explored Solved by Branching Strategy 23

Table 4.5 Low-Depth with Rounding: Problem Size vs. SDPs Solved by Branching Strategy 26

Table 4.6 Low-Depth with Rounding: Problem Size vs. Leaf Nodes Explored by Branching Strategy 27

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

BNB	branch and bound
DOD	Department of Defense
MI	mutual information
NPS	Naval Postgraduate School
PSD	positive semidefinite
SDP	semidefinite program
USN	U.S. Navy

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

The growth of China's Navy in the past few decades has been staggering. As a great power, China has the ability to significantly disrupt global trade and affect the economic interests of the United States. The U.S. Chief of Naval Operations, Admiral Michael Gilday, updated his NAVPLAN in 2022 in response to the latest National Defense Strategy, stating that the Navy must gain the decision advantage to operate in an environment advantageous to the United States. To do this we must invest in technology such as improved acoustic sensors in littoral regions.

In this thesis, we present research that aims to further the area of acoustic sensing. We present a branch and bound algorithm to solve the problem of finding the optimal placement for acoustic sensors in shallow waters. We use bathymetric and historical data from the Monterey Canyon around Monterey, California, as a proof of concept for our algorithm. The main goal of this thesis is to determine where to optimally place acoustic sensors in an area of interest if we are constrained to k sensors. We first introduce the military relevance for this problem and why it is necessary to continue research in this area. Then we look at relevant work that has already been done on spatial detection and explore the **mutual information** criterion as a basis for our problem formulation. Because finding the optimal placement for sensors is difficult we look at the celebrated **Goemans-Williamson** approximation algorithm as inspiration for a semidefinite program (SDP) relaxation to our optimization problem. We then propose a branch and bound algorithm to compute an optimal sensor placement. Finally, we look at two applications of our algorithm, one which involves randomly generated correlation matrices of varying size and a second which comes from the data we have on the Monterey Bay.

The algorithm [1] developed in this thesis has been made open source and publicly available on Github, where practitioners can further develop it for their own use. Recommendations for future work, and development of spatial detection and sensor placement beyond acoustic sensing are made as well.

References

[1] R. Bassett and E. Vargas, “Sensor placements which maximize mutual information,” GitHub, Aug. 21, 2022 [Online]. Available: https://github.com/rbassett3/mutual_info_sensor_placement

Acknowledgments

I would like to thank my parents for being my biggest cheerleaders throughout my life and a huge support to my career. To my sister, Emilee, thank you for always pushing me to reach for that which looks unobtainable and reminding me that our hard work pays off. Finally, I'd like to give appreciation to my wife, Amanda, for showing me what strength and tolerance are. Thank you for supporting me and my career and reminding me that there is always time for fun.

Special thanks to my amazing advising team, Dr. Robert Bassett, Dr. Jefferson Huang, and LTC Ross Schuchard, for mentoring me and encouraging me to learn new skills. They have truly made this thesis a project that I'm proud of and I've learned an immense amount from them.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Motivation

The current global environment is one where the United States is facing daily threats to democracy. While China acts as a pacing threat to the United States as a great power, we must utilize every avenue to ensure sea control and freedom of navigation. The chance of conflict arising in the Pacific is growing daily. The area surrounding China includes many natural and artificial islands that imply the need for shallow water operations. The Chief of Naval Operations and the Commandant of the Marine Corps have both indicated their willingness to increase Expeditionary Advanced Basing Operations and Littoral Operations in a Contested Environment to continue operating in an environment favorable to the United States [1]. With this in mind, the Navy needs to ensure it has systems in place to support increased amphibious operations in littoral environments.

Acoustic detection of ships and submarines has been an area of research for the Navy as far back as WWI. Antisubmarine warfare (ASW) often involves the implementation of at least one type of detection system. Contemporary methods for vessel detection include active and passive sonar as well as towed array sonar. While these are typically shipboard methods for detection, there are systems that allow remote detection capabilities. Sound Surveillance System (SOSUS) arrays and multistatic sonar systems are those which operate with remote placement of hydrophones or sonobuoys. With the need for littoral detection and island hopping coming to the forefront of warfare in the Pacific, shallow water sensing is becoming increasingly important. The CNO emphasized in his updated NAVPLAN 2022 that the Navy needs to increase Expeditionary Advanced Basing Operations (EABO) and Littoral Operations in a Contested Environment (LOCE). The key to doing this is by using acoustic sensors to get information about operating environments into the hands of decision makers to give them the decision advantage [1].

Placing acoustic sensors allows for remote detection of vessels in areas of interest. We study the problem of placing only a few sensors to effectively monitor signals at many locations.

Though our approach applies to sensor placement problems generally, we focus on an application placing hydrophones to efficiently monitor acoustic signals over a large region. Our starting point is the mutual information criteria for sensor placement, which despite being theoretically attractive has not been widely adopted because of its computational difficulty. To remedy these computational challenges, we introduce a novel branch and bound algorithm which relies upon a new semidefinite programming relaxation of the mutual information problem. Our contributions allow practitioners to solve sensor placement problems to global optimality while exploring only a fraction of the search space required by brute force.

1.2 Problem Introduction

In this thesis we consider the problem of optimally placing k sensors among n locations of interest in order to monitor a continuous measurement across the n locations. An optimal sensor placement is defined as one which *maximizes the mutual information* between locations with and without sensors. Measurements across the n locations are assumed to take the form of a Gaussian random vector with known covariance matrix, so that the measurement at each individual location is a Gaussian random variable. Our goal is to choose a set of placed sensors $A \subseteq \{1, \ldots, N\}$ with $|A| = k$ such that the mutual information between placed sensors and unplaced sensors is maximized.

This approach has some advantages over existing work in sensor placement problems. First, it avoids making strong assumptions on the relationship between sensors and the physical phenomenon of interest. For example, many papers assume that each sensor has a "detection window" $w : \mathbb{R}^d \to [0, 1]$ such that signal broadcast from position x_b with strength s is received at position x_r as the value $s \times w(x_b - x_r)$. Common examples include binary detection, where $w(x) = 1$ when $\|x\|$ is less than some fixed radius r and 0 otherwise. Other common window functions are explained in [2], [3], as well as slightly more complicated window functions, for example Cassini ovals, as in [4]. These "detection window" assumptions risk overly simplifying complex physical phenomena. This contrasts with our approach, which pairs well with using simulation designed to precisely model the physical phenomenon of interest, because these simulations can be used to precisely estimate the covariance matrix of the measurements. For example, in underwater acoustics, received signal strength

is a complicated function of a number of environmental characteristics which cannot be accurately modeled by a window function, but can be simulated using acoustic ray tracing models [5]. Additionally, though our motivation is primarily sensor placement in a spatial context, our formulation does not require any spatial embedding of the sensors or physical phenomenon of interest, which makes our results more general than those which require a spatial embedding. Finally, our use of mutual information as the sensor placement criterion has a natural information theoretic justification, while avoiding common limitations of other information-theoretic criteria for placing sensors, such as entropy.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces our main contribution, a novel semidefinite relaxation for the problem of maximizing mutual information with sensor placements. We review the mutual information criterion for sensor placement, while contrasting it with competing notions of optimal placements. It also briefly reviews relevant material on semidefinite relaxations of combinatorial optimization problems, which we will draw on in future sections. In Chapter 3, we incorporate this relaxation into a branch and bound implementation that finds optimal sensor placements effectively on large-scale problems. Numerical experiments are conducted in Chapter 4, where we apply our contributions to both simulated data and a hydrophone placement problem that involves tracking vessels off of the California coast.

Notation. Before proceeding we establish some notation. We denote a set of natural numbers $\{1, ..., n\}$ by $[n]$. Given an n dimensional vector X and $A \subseteq [n]$, we denote by X_A the subvector of X, of length $|A|$, which only contains entries X_i with $i \in A$. Similarly, $X_{[n]\setminus A}$ denotes the complementary subvector of X with entries X_i for $i \notin A$. Given an $n \times n$ matrix Σ, we let Σ_A be the square submatrix which has rows and columns from A and we let $\Sigma_{[n]\setminus A}$ be the square submatrix which has rows and columns from $[n] \setminus A$. We denote a matrix of 1s by $\mathbb{1}$. Finally, we use $\odot$ to indicate the Hadamard (elementwise) product between matrices.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Mutual Information for Sensor Placement

In this chapter, we review background material on mutual information and formalize the sensor placement problem with a mutual information criterion. We introduce the SDP relaxation that allows us to approximate solutions to our maximization problem in polynomial time. Finally, we discuss our method for rounding the solutions to the SDP relaxation to get a solution to the original objective function.

2.1 Mutual Information

Consider an n-dimensional Gaussian random vector X with covariance matrix Σ. We seek to find a subset $A \subseteq [n]$ of size k such that the measurements X_A are maximally informative. From an information-theoretic perspective, there are two natural criteria for formalizing the notation of "maximally informative" sensor placement. The first of these is minimizing the differential entropy of the random vector $X_{[n]\setminus A}$, given the measurements $X_A = x_A$. This is a measure of the uncertainty in the random vector $X_{[n]\setminus A}$ at unsensed locations conditioned upon the information we receive from the random vector X_A at sensed locations. For an n-dimensional Gaussian random vector with covariance Σ, this conditional differential entropy depends only on the covariance matrix Σ and not on the observation x_A, so we suppress x_A from our notation. Minimizing the differential entropy of $X_{[n]\setminus A}$ given X_A results in the following optimization problem[1], the solution of which is also called a D-optimal sensor placement.

$$\max_{A\subseteq[n],|A|=k} \operatorname{logdet}(\Sigma_A) \tag{2.1}$$

The logdet function is expensive to evaluate (requiring $k^3/2$ flops [6]) and difficult to optimize. Rewritten as a function of the eigenvalues $\lambda_1, \ldots \lambda_k$ of Σ_A, the problem in (2.1) maximizes $\sum_{i=1}^{k} \log \lambda_i$. Motivated by computational tractability, various alternatives to differential entropy have been proposed by considering different functions of the eigenvalues of Σ_A. In A-optimal sensor placement, we modify (2.1) to have objective $\operatorname{tr}(\Sigma_A) = \sum_{i=1}^{k} \lambda_i$.

[1]For completeness, we provide a proof of this result in Proposition 2 of the Appendix.

E-optimal sensor placement uses the objective $\lambda_{\min}(\Sigma_A)$, where $\lambda_{\min}$ denotes the minimum eigenvalue of Σ_A. Though A-optimal and E-optimal sensor placements can be easier to compute, they sacrifice the appealing information-theoretic justification that D-optimal sensor placement has through its connection to differential entropy [7].

Despite its intuitive formulation, D-optimal sensor placements have the drawback that in many applications they tend to place sensors on the boundary of an area of interest [8], [9], similar to the behavior that D-optimal designs exhibit in design of experiments [10]. This can be explained by noting that the differential entropy (up to constant factors) of $X_{[n]\setminus A}$ given X_A is $\operatorname{logdet}(\Sigma_A)$. Thus, the objective for D-optimal sensor placements does not explicitly model the relationship between the sensed vector X_A and the unsensed vector $X_{[n]\setminus A}$. To remedy this weakness, [8] proposes using the mutual information between X_A and $X_{[n]\setminus A}$ as an objective function instead of the differential entropy of $X_{[n]\setminus A}$ given X_A. From the perspective of sensor placement, the mutual information criterion improves upon the differential entropy criterion by including terms related to both X_A and $X_{[n]\setminus A}$.

In the following proposition, we recall the expression for mutual information between two complementary subvectors of a Gaussian random vector

Proposition 1 *Let $X \sim N(\mu, \Sigma)$ be a Gaussian random vector on $\mathbb{R}^N$. Given $A \subseteq [n]$, the mutual information between the vectors X_A and $X_{[n]\setminus A}$ is*

$$\frac{1}{2}\left(\operatorname{logdet}\Sigma_A + \operatorname{logdet}\Sigma_{[n]\setminus A} - \operatorname{logdet}\Sigma\right), \tag{2.2}$$

where Σ_A and $\Sigma_{[n]\setminus A}$ denote the covariance matrices of the random vectors X_A and $X_{[n]\setminus A}$, respectively.

For completeness, we include relevant definitions and a proof of Proposition 1 in the appendix.

By maximizing the mutual information between X_A and $X_{[n]\setminus A}$, we have a sensor placement criterion with a clear information theoretic justification which improves upon the drawbacks of differential entropy. Neglecting constant terms in (2.2), we arrive at the following optimization problem for placing sensors that maximize the mutual information between

sensed and unsensed locations' variables.

$$\max_{\substack{A\subseteq[n]\\|A|=k}} \operatorname{logdet}\Sigma_A + \operatorname{logdet}\Sigma_{[n]\setminus A} - \operatorname{logdet}\Sigma, \tag{2.3}$$

Problem (2.3) will be our focus for the remainder, where we will investigate new approaches for provably solving it to global optimality.

Our focus on global optimality complements that of [9], where the authors develop approximate solutions to (2.3). They focus on solving a similar problem of choosing sensor locations for detecting spatial phenomena. They build off of [8], using the entropy criterion as a starting point to solving the problem of best sensor placement. As maximizing mutual information is an NP-complete problem, they develop an approximation algorithm to solve it. Their approximation algorithm is greedy where they add sensors in sequence such that the increase in mutual information is maximized.

Our approach to solving the sensor placement problem is through a relaxation where we reformulate the optimization problem as a semidefinite program (SDP). We iteratively make sensor placements and solve SDPs by a branch and bound algorithm until k sensors have been placed. By using an SDP relaxation the relaxed problem becomes solvable in polynomial time and we recover solutions to the original problem using a randomized rounding scheme.

2.2 Semidefinite Relaxations of Combinatorial Optimization Problems

A discussion of the Max-Cut problem in network theory is necessary to understand why we use a relaxation scheme similar to the celebrated *Goemans-Williamson* relaxation [11]. Suppose we have a graph $G = (V, E)$ with vertices V and edges E where each edge has a weight. Further, we want to create a cut of G that partitions it into two subsets, S and $V \setminus S$ in such a way that the weight between the bisected edges is maximized. This is the Max-Cut problem which, in itself, has been shown to be NP-complete [11]. The optimization problem of Max-Cut is stated as follows

$$\max \sum_{\{i,j\}\in E} \left(A \odot \frac{\mathbb{1} - \mathbf{u}\mathbf{u}^T}{2} \right)_{ij}, \tag{2.4}$$

where A is the adjacency matrix of the graph G and $\mathbf{u} \in \{-1, 1\}^n$ gives the partition of the nodes in G. That is, $\mathbf{u}_i = 1$ if node $i \in S$ and $\mathbf{u}_i = -1$ if node $i \in V \setminus S$. This problem formulation is NP-complete, so to solve the Max-Cut problem, a relaxation is introduced. By letting $\mathbf{X} = \mathbf{u}\mathbf{u}^T$, which is positive semidefinite[2] (PSD) we can reformulate the problem as

$$\begin{aligned} \max_{\mathbf{X} \succeq 0} \ & \sum_{i,j} \left(A \odot \frac{\mathbb{1} - \mathbf{X}}{2} \right)_{i,j} \\ \text{s.t.} \quad & \operatorname{diag}(\mathbf{X}) = 1 \\ & \operatorname{rank}(\mathbf{X}) = 1. \end{aligned} \tag{2.5}$$

Finally, relaxing the problem by dropping the rank constraint, the relaxed Max-Cut problem is

$$\begin{aligned} \max_{\mathbf{X} \succeq 0} \ & \sum_{i,j} \left(A \odot \frac{\mathbb{1} - \mathbf{X}}{2} \right)_{i,j} \\ \text{s.t.} \quad & \operatorname{diag}(\mathbf{X}) = 1. \end{aligned} \tag{2.6}$$

The Goemans-Williamson approximation algorithm is then, solve problem (2.6), which is a semidefinite program, then use randomized rounding to recover a solution to the original unrelaxed problem.

2.3 SDP Relaxation of Maximizing Mutual Information

The problem of finding the optimal placement of sensors with constraints is an NP-hard problem [9]. Therefore, we use two reformulations of our problem and a relaxation similar to *Goemans-Williamson* approximation algorithm to obtain results.

We've shown in (2.3) that our problem formulation and objective function is

$$\max_{\substack{A \subseteq [n] \\ |A| = k}} \operatorname{logdet} \Sigma_A + \operatorname{logdet} \Sigma_{[n] \setminus A} - \operatorname{logdet} \Sigma.$$

[2]Recall that $\mathbf{X}$ is a positive semidefinite (PSD) matrix if and only if it has a Cholesky factorization, $\mathbf{X} = \mathbf{U}\mathbf{U}^T$ for some matrix $\mathbf{U}$ [11].

From this function we reformulate it into an equivalent function then apply a relaxation. Notice that this formulation is similar to that in (2.4).

$$\max_{\mathbf{u}\in\{-1,1\}^n} \log\det\left(\Sigma \odot \frac{\mathbf{u}\mathbf{u}^T + \mathbb{1}}{2}\right) - \log\det(\Sigma) \tag{2.7}$$

The vector $\mathbf{u}$ represents a vector of length n of -1s and 1s where the i^{th} entry is 1 if a sensor was placed at that location and -1 if it was not. Taking the outer product $\mathbf{u}\mathbf{u}^T$ and adding a matrix of 1s, we end up with 2s and 0s, and thus the final matrix $\frac{\mathbf{u}\mathbf{u}^T+\mathbb{1}}{2}$ is a matrix of 1s and 0s. This, along with the Hadamard product of the covariance matrix Σ, encodes the data about the region of interest to the maximization problem.

Finally, we utilize a formulation similar to that of the Max-Cut formulation in (2.5). First, define X_{ij} as $\mathbf{u}\mathbf{u}^T$. Then, $X_{ij} = 1 \iff u_i = 1, u_j = -1$ or $u_i = -1, u_j = 1$. Next, let $Q = \Sigma \odot \frac{\mathbf{u}\mathbf{u}^T+\mathbb{1}}{2}$ We then apply permutation matrix P to Q; this matrix is PQP^T. The permutation matrix is orthogonal so conjugating Q by it does not change its determinant. Finally, PQP^T is block diagonal so the determinant is a product of the determinants of its diagonal blocks.

One further reformulation to an equivalent problem allows us to use a similar relaxation to that of *Goemans-Williamson*,

$$\begin{aligned} \max_{\mathbf{X}\geq 0} \ & \log\det\left(\Sigma \odot \frac{\mathbf{X} + \mathbb{1}}{2}\right) - \log\det(\Sigma) \\ \text{s.t.} \quad & \operatorname{diag}(\mathbf{X}) = 1 \\ & \operatorname{rank}(\mathbf{X}) = 1. \end{aligned} \tag{2.8}$$

The formulation of the Max-Cut problem is indeed very similar to the formulation of our sensor placement problem, so our relaxation is inspired by *Goemans-Williamson*. We get rid of the $\operatorname{rank}(\mathbf{X}) = 1$ constraint and have our final objective value function

$$\max_{\mathbf{X}\succeq 0} \ \log\det\left(\Sigma \odot \frac{\mathbf{X}+\mathbb{1}}{2}\right) \quad \text{s.t.} \quad \operatorname{diag}(\mathbf{X}) = 1. \tag{2.9}$$

This relaxed optimization problem is now a convex optimization problem and solvable in polynomial time [11]. To show this, note that the logdet function is concave [12], $\mathbf{X} \succeq 0$ is known to be a convex set, $\operatorname{diag}(\mathbf{X}) = 1$ is an affine constraint, and the operations on our set of interest, $\mathbf{X} \succeq 0$, are linear operations. Once we solve this relaxed problem we use a randomized rounding scheme outlined in the next subsection.

2.3.1 Outward Rotation Rounding

To recover a solution to our original problem, we use a randomized rounding scheme called *outward rotations* outlined in [13]. First, let $v_1, v_2, \ldots, v_n$ be unit vectors which are the solution to our SDP relaxation. Further, let $u_1, u_2, \ldots, u_n$ be a set of orthonormal vectors which are also orthogonal to the vectors $v_1, v_2, \ldots, v_n$. Choose an angle, γ, such that $0 \leq \gamma \leq \pi/2$. Then each vector v_i can be rotated by angle γ towards u_i in the plane spanned by v_i and u_i. The resultant rotated vectors, $v_1', v_2', \ldots v_n'$, can then be rounded using a random hyperplane. Letting γ vary between 0 and $\pi/2$, we obtain a combination of random hyperplane rounding and independent random choices. In our implementation, γ is chosen at evenly spaced values in the interval $0 \leq \gamma \leq \pi/2$ where the total number of values chosen is the number of roundings conducted.

CHAPTER 3:
Branch and Bound Algorithm

This chapter discusses our development of a branch and bound algorithm which solves the global problem of maximizing mutual information. Using this algorithm, we iteratively decide whether to place or exclude a sensor from a candidate sensor location. We first introduce the partially determined subproblem which includes currently placed sensors when descending down the tree. Then, we discuss our methods for obtaining upper and lower bounds on the optimal objective value and look at how to trim away suboptimal portions of the tree. The benefit of using this branch and bound method is that we do not need to explore the entire tree to obtain an optimal sensor placement.

The goal with our branch and bound is to find the vector $\mathbf{u} \in \{-1, 1\}^n$ with k 1s where n is the number of candidate sensor locations, k is the number of sensors to place, and $\mathbf{u}_i = 1$ if a sensor was placed at location i and $\mathbf{u}_i = -1$ if a sensor was not placed at location i. We start by initializing the vector $\mathbf{u} = \mathbf{0}$ where $\mathbf{u}_i = 0$ if location i is undecided for sensor placement. From here, we can begin branching based on locations that have not been explored by solving *partially determined subproblems*.

3.1 Partially Determined Subproblem

At each candidate location, i, we must decide whether to place a sensor (set $\mathbf{u}_i = 1$) or to not place a sensor (set $\mathbf{u}_i = -1$). The partial solution to the overall problem of finding the optimal placement of k sensors is a vector $\mathbf{u} \in \{-1, 0, 1\}^n$. Given the partially determined $\mathbf{u}$, we choose an index i such that $\mathbf{u}_i = 0$ and solve two SDP relaxations for $\mathbf{u}_i = 1$ and $\mathbf{u}_i = -1$. The solutions to these two SDP solves provide us with upper bounds on the optimal objective value with the current partial information in the scenarios where we place a sensor at location i or we do not. The formulation of the partially determined maximization problem follows from (2.9) where we have the following equation,

$$
\begin{aligned}
\max_{\mathbf{X}\succeq 0} \quad & \log\det\left(\Sigma \odot \frac{\mathbf{X}+\mathbb{1}}{2}\right) \\
\text{s.t.} \quad & \operatorname{diag}(\mathbf{X}) = 1 \\
& \mathbf{X}_{ij} = \mathbf{u}_i\mathbf{u}_j \ \text{ if } \ \mathbf{u}_i, \mathbf{u}_j \neq 0 \\
& \sum_{j=1}^{n} \mathbf{X}_{i,j} = \begin{cases} n-2k & \text{if } \mathbf{u}_i = -1 \\ 2k-n & \text{if } \mathbf{u}_i = 1 \end{cases}
\end{aligned}
\tag{3.1}
$$

The two additional constraints are in support of having partial information on candidate senor locations where we have decided to place or not place a sensor. The first constraint specifies that the ij^{th} entry of $\mathbf{X}$ will be equal to $\mathbf{u}_i\mathbf{u}_j$ if a decision about placing a sensor at the i^{th} and j^{th} location has already been made (i.e., $\mathbf{u}_i, \mathbf{u}_j \neq 0$) and $i \neq j$. The second constraint specifies the row sums of the matrix $\mathbf{X}$. Given we are placing k sensors and choosing not to place sensors in $n-k$ locations, this constraint uses already obtained information based on previous decisions to limit the i^{th} row sum dependent on the value of $\mathbf{u}_i$. These constraints also induce column sum constraints since $\mathbf{X}$ is symmetric.

3.2 Upper and Lower Bounds

The lower bounds on the optimal objective value to our maximization problem (2.9) are obtained using heuristics and rounding. We initialize the lower bound by using a greedy algorithm to place sensors. Descending down the tree, each solution to (3.1) gives us upper bounds. These upper bounds, however, are local to the current branch we are exploring with the given partial information. We then use outward rotations 2.3.1 to round the SDP maximizer, $\mathbf{X}$, to get a feasible solution to our original problem which also provides us with updated global lower bounds to the optimal objective value. Finally, if we descend to the bottom of a branch to a leaf node, we also update our lower bounds from that solution. If at any point in the tree the global lower bound exceeds the upper bound at that node, we do not need to continue exploring the current branch because any remaining solutions in that branch will be suboptimal. The benefit of this is that we can reduce the number of leaf nodes we need to explore because we are not exploring the entire tree. Once we cut a branch, we restart solving partially determined subproblems.

3.3 Implementation in Low-Level Languages

The implementation of this thesis has been primarily done in Python. However, because of the computational complexity of calculating the logdet and evaluating SDPs in the branch and bound, Python is not ideal for creating a fast solver. Therefore, we've used C++ for the branch and bound framework to include: initializing the problem, updating upper and lower bounds, branching, and terminating a branch. We used FORTRAN and the Splitting Conic Solver [14] to handle the mathematical computations, specifically: the greedy algorithm for placing sensors, calculating logdet, evaluating SDP objective values, and performing outerward rotation roundings as specified in 2.3.1. These solutions were then utilized by C++ to iterate through the branch and bound tree. With the majority of mathematical computations being done in low-level languages, we utilize Python as a wrapper that the user can interface with. The algorithm utilizes a priority queue to keep track of problems which still need to be solved. That is, the candidate sensor locations that have not been explored for sensor placement are put into a queue. To decide which problem (node) to explore, each unexplored location is ranked based on its upper bounds. From here, the problem with the maximum upper bound is evaluated first in hopes that solving this problem will lead to updating our lower bound. The contents of this algorithm have been made open source and hosted on GitHub [15] at https://github.com/rbassett3/mutual_info_sensor_placement.

Another important variation we explore is the *branching strategy* which determines the criteria for deciding to place or not place a sensors. There are three criteria we implement to examine the variation in performance. The first is *greedy good* where the algorithm first chooses a variable that represents an unplaced sensor. We take the partial information for the current branch and complete it by assigning a sensor to an undecided location, i.e., we assign a 1. Then, with the remaining undecided locations we assign -1s and solve for the objective function value. We continue placing sensors in undecided locations in this way with the local partial information, finally deciding which sensor to place based on the best (greatest) objective function value. The second branching strategy is *greedy bad* where we perform the same procedure as in *greedy good* but we pick the worst (lowest) objective function value. The last method is *random* selection of the branching variable, regardless of the feasible solution value. We used these strategies to compare the performance of the three experiments in Chapter 4.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Computational Experiments

In this chapter, we outline the methods for experimentation and comparison of results for our branch and bound algorithm. So far, we have discussed the theory behind solving the SDP relaxation and the implementation of our branching algorithm to optimally select sensor placements. However, to implement this algorithm there are several parameters that we need to account for, each of which will influence how the algorithm solves the SDPs and descends through each tree. We explore how changing the depth of the branching tree affects computation time and efficiency for obtaining solutions. First we review a method for generating random correlation matrices called the *onion method*. We use the randomly generated matrices of varying size to compare different methods for choosing branching variables. Then, we present a proof-of-concept application of placing sensors in the Monterey Bay with given bathymetric and historical data of the Monterey Canyon.

4.1 Onion Method

For a d-dimensional random vector X, let $\mathbf{\Sigma}_X$ denote its covariance matrix. The *correlation matrix* associated with X is

$$\text{corr}(X) = \text{diag}(\mathbf{\Sigma}_X)^{-1/2}\mathbf{\Sigma}_X\text{diag}(\mathbf{\Sigma}_X)^{-1/2}.$$

A necessary and sufficient condition for a matrix $\mathbf{S}$ to be the correlation matrix of some random vector is that $\mathbf{S}$ be symmetric, positive semidefinite, and have diagonal entries identically equal to 1. Let Ω denote the set of all correlation matrices.

As part of their analysis of the NORTA Method for generating correlated random vectors with given marginal distributions and a given correlation matrix, [16] proposed the "onion method" for sampling uniformly from Ω:

1. Set $\mathbf{S}_1$ as the 1×1 matrix containing the element 1.
2. For $k = 2, \ldots, d$,

(a) Sample a "radius" $R \sim \text{Beta}\left(\frac{k-1}{2}, \frac{d-k}{2} + 1\right)$

(b) Sample a unit vector $\Theta \in \mathbb{R}^{k-1}$ uniformly from the surface of the $(k-1)$-dimensional sphere.

(c) Set $Q = \mathbf{S}_{k-1}^{1/2} r\Theta$.

(d) Set $\mathbf{S}_k = \begin{bmatrix} \mathbf{S}_{k-1} & Q \\ Q^T & 1 \end{bmatrix}$

Many efficient methods exist for sampling from univariate Beta distributions (see e.g., [17]). Moreover, sampling a point that is uniformly distributed on an n-dimensional sphere can be accomplished by sampling a random vector $V \in \mathbb{R}^n$ whose coordinates are independent standard normal random variables, and then setting $\Theta = \frac{V}{\|V\|_2}$.

4.2 Experiments

To test the effectiveness of the branch and bound algorithm, we set up experiments to compare our technique with other branching techniques. Our experiments were performed on a dataset that was created using the *onion method* outlined in Section 4.1. The onion method generates random correlation matrices that we used to test the different branching schemes. The data consists of correlation matrices that span in size from 2×2 to 25×25 with each size of matrix having 100 different observations on which we ran our branch and bound algorithm.

At each SDP solve, the branching algorithm is given a choice of which variable (sensor location) to branch on as discussed in 3.3. We implement three branching strategies that the algorithm uses to make a choice, *greedy good*, *greedy bad*, and *random*. The metric for comparison was number of leaf nodes explored and number of SDPs solved. This data was compared with the total number of leaf nodes that could be explored if brute force had been used to find the optimal placement.

4.2.1 Implementation Parameters

The branch and bound implementation has multiple parameters that can be changed which affects the way the tree is explored. To make an effective comparison of different branching techniques it is necessary to discuss the important parameters that vary between our experiments. First is the number of candidate sensor locations, n. This is determined based off of

the size of correlation matrices where the matrices are $n \times n$. Next is the number of sensors we wish to place, k. For our experiments, this has been held constant at $\lfloor \frac{n}{2} \rfloor$. Tolerance is the next parameter, which indicates the numerical tolerance for each SDP solve. The next parameter is *depth* which is the number of sensor placements to assign before the algorithm reverts to brute force. It is important to note that a depth of 0 would be indicative of solely relying on brute force to make assignments. This would result in a tree with $\binom{n}{k}$ leaf nodes. Next is the number of random roundings to perform from the relaxed SDP solution. This is accomplished using the outward rotations method described in Section 2.3.1. These parameters, along with the three branching strategies described in 3.3, were varied throughout our experiments to determine their effects on the number of SDPs solved and the number of leaf nodes explored.

4.2.2 Full-Depth Tree Exploration without Rounding

In this first experiment, we look at our baseline case where we set depth = n (i.e., the size of the covariance matrix). This means that during the process of calculating SDPs, k sensors will be placed and brute force will not be used to make any sensor placements. Tolerance was set to .01 and number of rounds was set to 0 for this experiment. Tables 4.1 and 4.2 show how the average number of SDPs solved and the average number of leaf nodes explored changes by branching strategy and problem size, respectively. Further, we juxtapose the total number of potential leaf nodes that could be explored based on problem size and number of sensors to place. In the smallest correlation matrices, the three branching strategies perform equally in terms of SDPs solved and leaf nodes explored, likely due to the small problem sizes. We display the number of leaf nodes explored in Figure 4.1 where we have scaled the y-axis logarithmically. An analysis of the number of leaf nodes explored by the branch and bound algorithm shows how efficient the *greedy good* method is for solving our problem. When analyzing larger matrices, we can see how our algorithm greatly reduces the number of leaf nodes that need to be explored. *Greedy good* performs quite well when compared to *greedy bad* and *random*. Of note, after size 17 matrices, the *random* method does as well or better than *greedy good*. That is, the method where an unexplored sensor placement is chosen at random instead of based on our *greedy good* strategy shows more leaf nodes being cut away from the tree. During the course of the experiment, our algorithm was able to evaluate through matrices of size 22, which will be contrasted with the experiments where

we introduce rounding and low depth.

Table 4.1. Full-Depth without Rounding: Problem Size vs. SDPs Solved by Branching Strategy

Problem Size	Greedy Good	Greedy Bad	Random
2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
4	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
5	3.8 ± 0.6	3.96 ± 0.28	3.9 ± 0.44
6	9.58 ± 0.82	9.9 ± 0.59	9.84 ± 0.61
7	13.26 ± 2.87	17.42 ± 1.4	16.44 ± 2.44
8	28.5 ± 5.59	35.42 ± 3.85	33.58 ± 4.74
9	41.3 ± 11.81	62.78 ± 7.0	56.0 ± 10.11
10	87.4 ± 21.02	118.96 ± 17.26	109.68 ± 18.12
11	115.58 ± 34.37	204.28 ± 35.97	175.32 ± 35.26
12	264.2 ± 77.77	392.42 ± 76.09	348.46 ± 72.99
13	360.56 ± 133.2	648.16 ± 180.65	557.22 ± 161.38
14	696.38 ± 266.17	1158.68 ± 323.42	987.62 ± 271.56
15	1012.66 ± 355.6	2028.88 ± 582.18	1629.58 ± 452.38
16	2279.32 ± 801.85	3877.08 ± 1147.07	3322.3 ± 936.87
17	2962.82 ± 1093.66	6269.68 ± 2132.39	4987.74 ± 1547.64
18	6109.52 ± 1864.06	11393.32 ± 3429.19	9393.98 ± 2554.08
19	9338.0 ± 3480.67	19747.42 ± 6801.35	15903.76 ± 5148.36
20	20191.78 ± 6568.56	37383.96 ± 12064.33	30958.82 ± 9256.69
21	28431.86 ± 9992.66	62457.56 ± 18927.22	48609.6 ± 14478.51
22	59927.8 ± 20436.18	111064.42 ± 39315.03	91070.64 ± 28814.44

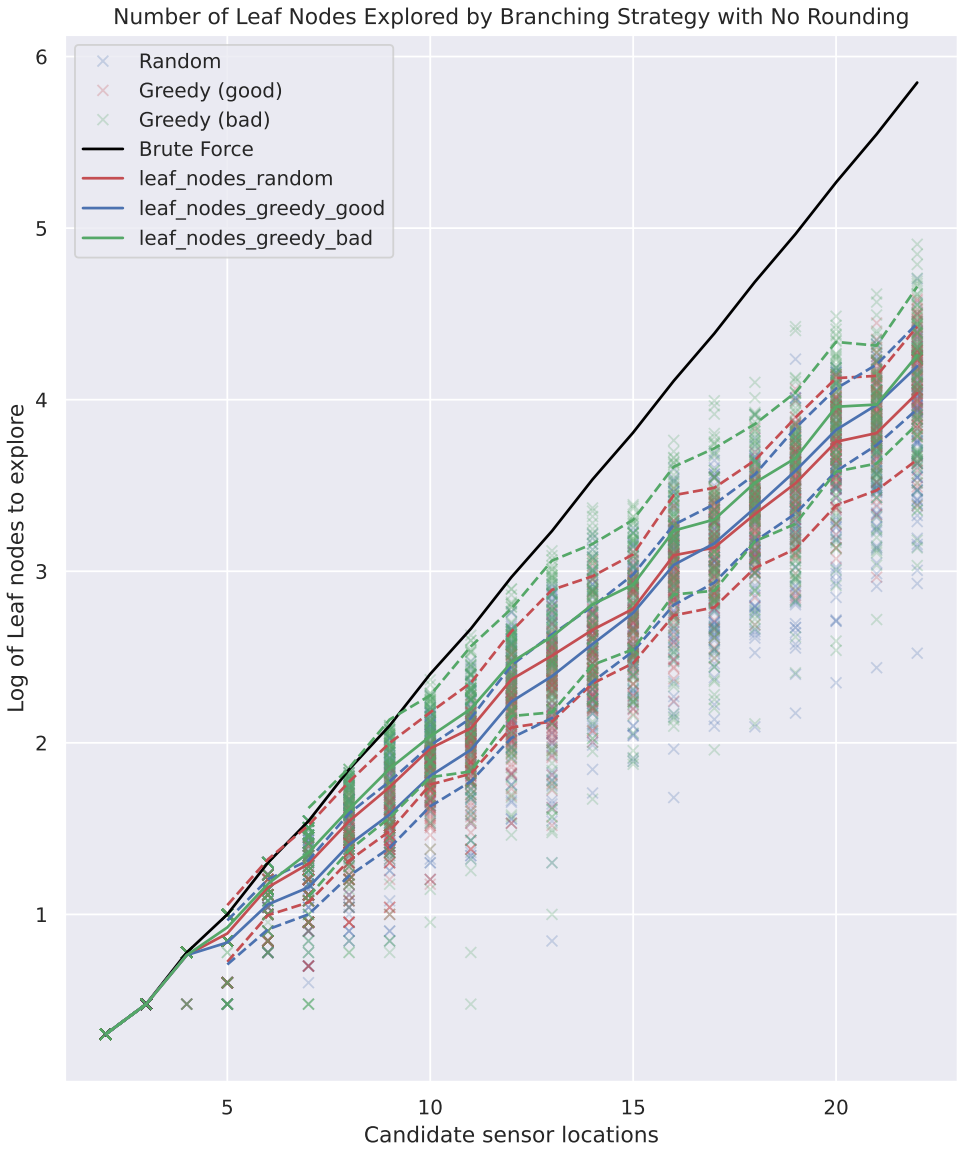
Mean number of SDP solves ± 1 standard deviation over 100 solves

Table 4.2. Full-Depth without Rounding: Problem Size vs. Leaf Nodes Explored by Branching Strategy

Problem Size	Greedy Good	Greedy Bad	Random	Brute Force
2	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2
3	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	3
4	5.79 ± 0.98	5.79 ± 0.98	5.79 ± 0.98	6
5	6.86 ± 1.84	8.4 ± 2.35	7.74 ± 2.42	10
6	11.44 ± 3.64	15.06 ± 4.78	14.39 ± 4.57	20
7	14.37 ± 4.88	22.95 ± 9.46	19.71 ± 8.25	35
8	25.49 ± 10.19	41.05 ± 17.55	34.95 ± 14.34	70
9	38.19 ± 15.75	70.78 ± 33.15	55.24 ± 27.03	126
10	64.17 ± 22.74	109.18 ± 48.26	92.7 ± 36.98	252
11	91.0 ± 33.65	157.63 ± 95.03	121.56 ± 63.61	462
12	173.84 ± 76.15	294.34 ± 175.04	234.09 ± 128.53	924
13	244.49 ± 118.08	416.78 ± 320.98	322.16 ± 218.68	1716
14	376.41 ± 208.01	640.84 ± 505.77	455.5 ± 340.73	3432
15	571.61 ± 272.13	835.8 ± 619.14	604.38 ± 409.12	6435
16	1089.29 ± 532.66	1725.42 ± 1216.53	1238.58 ± 815.53	12870
17	1453.38 ± 720.8	2002.29 ± 1915.83	1373.86 ± 1028.49	24310
18	2340.14 ± 1004.67	3283.8 ± 2367.55	2152.85 ± 1417.57	48620
19	3849.68 ± 2014.25	4562.68 ± 4325.18	3257.65 ± 2684.77	92378
20	6666.75 ± 3204.82	9096.15 ± 6416.73	5684.92 ± 3862.56	184756
21	9337.74 ± 4676.29	9366.85 ± 7256.65	6389.16 ± 4778.39	352716
22	15654.52 ± 8054.41	18157.44 ± 15324.52	10927.18 ± 8519.1	705432

Mean number of leaf nodes explored ± 1 standard deviation over 100 solves

Figure 4.1. Comparison of Branching Methods with Full Depth and No Rounding



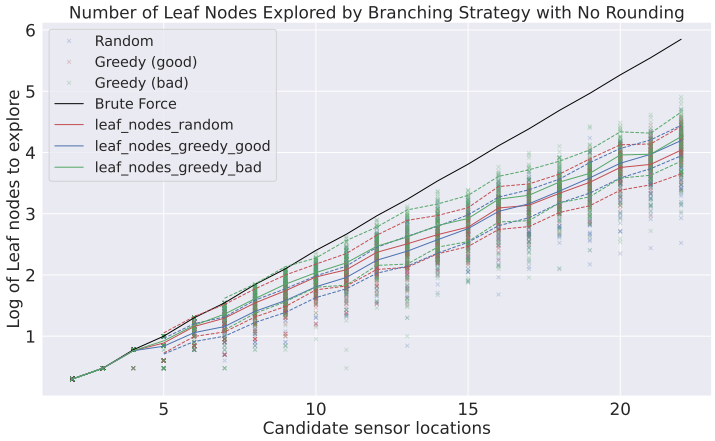
Solid lines indicate the mean and dotted lines indicate ± 1 standard deviation

4.2.3 Full-Depth Tree Exploration with Rounding

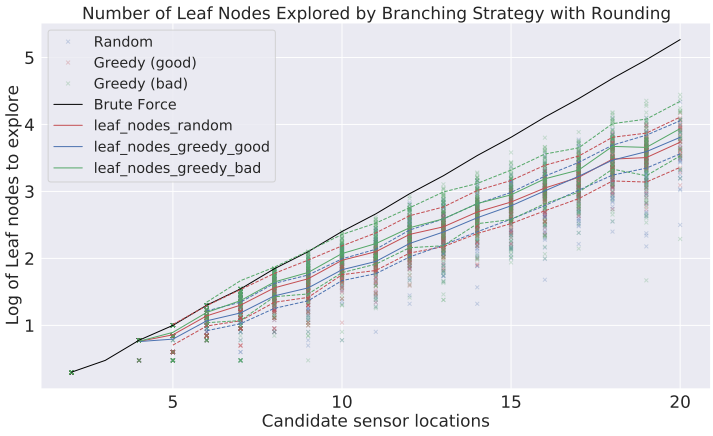
In this experiment, we set depth = n, the number of sensors to place was k, and tolerance was .01 as in 4.2.2. For this experiment there were $500 \cdot \lfloor \frac{n}{2} \rfloor$ random roundings done on the SDP solutions. The results from this experiment are contrasted in Figure 4.2 with those from the baseline experiment shown in Figure 4.1.

Figure 4.2. Difference between Rounding and Not Rounding

(a) The baseline experiment with no rounding



(b) The experiment with rounding included



Solid lines indicate the mean and dotted lines indicate ± 1 standard deviation

The results of number of SDPs solved and number of leaf nodes explored are shown in Tables 4.3 and 4.4, respectively. We see very little difference when contrasting the two experiments with each other. The *random* branching strategy still begins to perform better than *greedy good* when evaluating correlation matrices of size 17 and larger (in terms of number of leaf nodes explored). Whereas previously we assumed that rounding our SDP solutions would improve the performance of the branch and bound by having to explore less of the tree, this does not seem to be the case. Additionally, this experiment was only able to evaluate up to size 20 matrices due to the intensity of computation when performing 500 roundings on each matrix.

Table 4.3. Full-Depth with Rounding: Problem Size vs. SDPs Solved by Branching Strategy

Problem Size	Greedy Good	Greedy Bad	Random
2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
4	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
5	3.68 ± 0.74	3.98 ± 0.2	3.82 ± 0.58
6	9.68 ± 0.79	9.98 ± 0.2	9.8 ± 0.72
7	13.28 ± 3.11	17.18 ± 1.64	16.4 ± 2.4
8	29.8 ± 5.64	35.62 ± 3.87	34.52 ± 4.25
9	39.08 ± 11.21	60.22 ± 9.14	54.64 ± 10.75
10	89.68 ± 17.65	120.92 ± 17.33	110.74 ± 15.9
11	119.06 ± 34.35	203.22 ± 38.27	178.74 ± 34.49
12	255.92 ± 79.24	378.42 ± 78.85	341.42 ± 74.84
13	358.04 ± 115.44	647.18 ± 160.88	550.18 ± 130.05
14	725.4 ± 238.38	1196.24 ± 290.36	1040.78 ± 259.07
15	1038.91 ± 349.21	2058.74 ± 556.11	1728.34 ± 472.3
16	2169.0 ± 695.81	3698.26 ± 1110.11	3182.32 ± 856.53
17	3281.14 ± 1101.78	6683.54 ± 1794.85	5421.88 ± 1496.85
18	7118.0 ± 2232.83	13108.57 ± 3764.16	10704.87 ± 2857.44
19	9557.24 ± 3361.98	20154.56 ± 6920.41	16056.32 ± 5108.46
20	19684.06 ± 7672.96	37529.5 ± 12953.91	30134.31 ± 10420.32

Mean number of SDP solves ± 1 standard deviation over 100 solves

Table 4.4. Full-Depth with Rounding: Problem Size vs. Leaf Nodes Explored Solved by Branching Strategy

Problem Size	Greedy Good	Greedy Bad	Random	Brute Force
2	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2
4	5.73 ± 1.05	5.79 ± 0.88	5.76 ± 0.92	6
5	6.23 ± 1.91	7.86 ± 2.93	7.23 ± 2.25	10
6	11.69 ± 3.69	15.6 ± 4.5	13.88 ± 4.41	20
7	15.33 ± 5.18	23.42 ± 10.0	20.03 ± 8.4	35
8	27.58 ± 10.96	44.48 ± 17.94	36.31 ± 15.18	70
9	36.26 ± 14.35	61.63 ± 32.33	49.67 ± 25.04	126
10	67.68 ± 22.42	118.03 ± 55.61	93.7 ± 35.79	252
11	89.96 ± 32.7	166.14 ± 98.13	125.83 ± 64.63	462
12	166.93 ± 78.98	286.94 ± 176.26	228.2 ± 135.29	924
13	247.96 ± 105.73	389.52 ± 292.7	296.13 ± 178.32	1716
14	404.48 ± 185.22	658.34 ± 427.77	492.81 ± 306.57	3432
15	610.2 ± 265.21	889.62 ± 705.16	694.4 ± 460.54	6435
16	1019.12 ± 458.2	1542.5 ± 1194.37	1124.27 ± 745.47	12870
17	1682.11 ± 762.79	2086.6 ± 1447.16	1622.96 ± 1139.07	24310
18	2931.36 ± 1469.33	4700.95 ± 3421.31	3035.71 ± 2121.38	48620
19	3924.99 ± 1897.57	4541.8 ± 4082.97	3199.82 ± 2387.43	92378
20	6424.72 ± 3325.63	8666.22 ± 6506.96	5428.75 ± 4435.36	184756

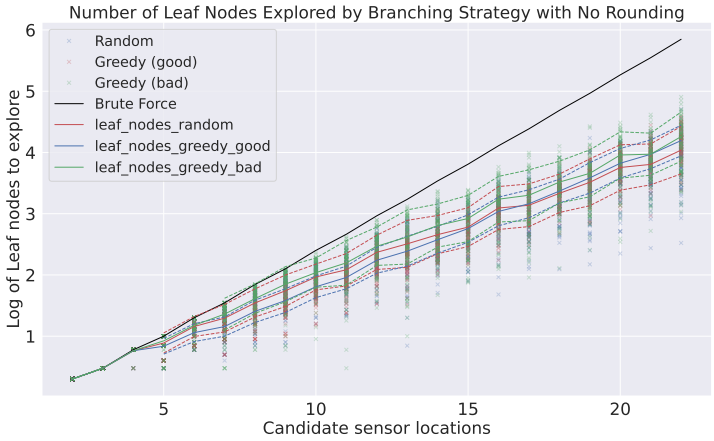
Mean number of leaf nodes explored ± 1 standard deviation over 100 solves

4.2.4 Low-Depth Tree Exploration

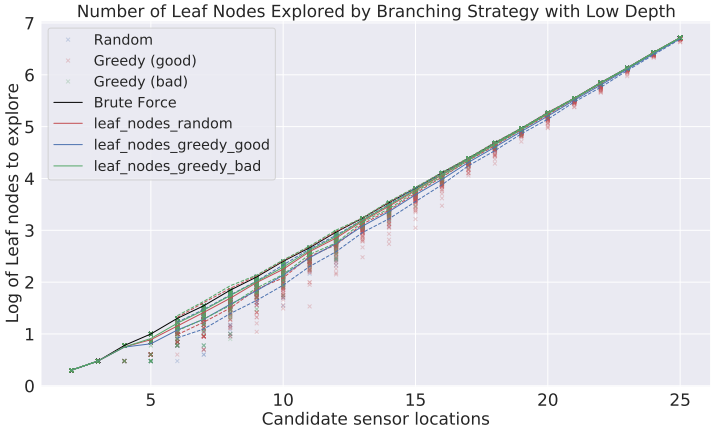
For this experiment we set depth = $\lfloor \frac{n}{2} \rfloor$ so that after half of candidate sensor locations have been explored and decided, the algorithm will resort to brute force to explore and decide on placements for the remaining locations. For each matrix, there were $500 \cdot \lfloor \frac{n}{2} \rfloor$ random roundings done. The results from this experiment are contrasted similarly in Figure 4.3.

Figure 4.3. Difference between Full Depth and Low Depth

(a) The baseline experiment with full depth



(b) The experiment with low depth



Solid lines indicate the mean and dotted lines indicate ± 1 standard deviation

The results of number of SDPs solved and number of leaf nodes explored are shown in Tables 4.5 and 4.6, respectively. This experiment shows a stark difference to the full-depth experiment where we see many leaf nodes being explored. An analysis of Table 4.6 supports this by showing how varying the problem depth affects the number of leaf nodes explored. For very small matrices, size < 6, it seems that *greedy good* does perform well, but with problems of this size, all branching methods perform similarly. When problem size increases there is decreasing gap between all branching methods and brute force. That is, as problem size increases and low depth is specified, each branching method performs almost as poorly as brute force. With the depth set so half of the candidate locations are explored, the algorithm tends to explore almost all possible leaf nodes or combinations of placements when problem size is large. Of note, this method was able to evaluate all matrices up through size 25. This is valuable to note because in the previous experiments, we were able to reduce the number of leaf nodes being explored significantly, but problem evaluation took considerably longer. In this experiment with low depth, many leaf nodes were explored but the problems were evaluated much more quickly.

Table 4.5. Low-Depth with Rounding: Problem Size vs. SDPs Solved by Branching Strategy

Problem Size	Greedy Good	Greedy Bad	Random
2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
4	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
5	3.74 ± 0.68	3.94 ± 0.34	3.9 ± 0.44
6	9.7 ± 0.77	9.96 ± 0.4	9.88 ± 0.62
7	10.74 ± 1.29	11.82 ± 0.7	11.5 ± 1.0
8	22.62 ± 2.66	25.38 ± 1.79	24.46 ± 2.29
9	24.28 ± 2.49	27.36 ± 1.82	27.18 ± 1.68
10	49.32 ± 7.04	55.72 ± 4.12	54.6 ± 4.73
11	53.86 ± 5.69	59.1 ± 2.19	58.36 ± 3.16
12	105.84 ± 12.28	117.38 ± 9.39	115.48 ± 9.96
13	113.52 ± 9.59	121.98 ± 3.89	121.16 ± 4.73
14	219.22 ± 23.68	244.32 ± 9.57	241.12 ± 11.82
15	233.48 ± 20.23	248.96 ± 7.1	247.14 ± 7.82
16	457.38 ± 38.04	496.5 ± 13.81	495.92 ± 12.39
17	486.02 ± 22.07	504.0 ± 7.65	504.2 ± 7.42
18	949.02 ± 57.08	1008.36 ± 15.07	1004.94 ± 21.31
19	985.42 ± 30.44	1011.58 ± 10.12	1013.14 ± 10.06
20	1923.0 ± 110.56	2026.48 ± 19.52	2019.56 ± 35.73
21	1996.78 ± 49.03	2034.12 ± 11.28	2037.48 ± 9.22
22	3914.32 ± 162.58	4065.66 ± 26.29	4069.8 ± 25.51
23	4028.02 ± 77.62	4074.68 ± 19.77	4083.38 ± 16.57
24	7976.34 ± 160.54	8155.44 ± 30.11	8167.51 ± 16.36
25	8106.85 ± 94.55	8169.64 ± 18.19	8179.92 ± 13.96

Mean number of SDP solves ± 1 standard deviation over 100 solves

Table 4.6. Low-Depth with Rounding: Problem Size vs. Leaf Nodes Explored by Branching Strategy

Problem Size	Greedy Good	Greedy Bad	Random	Brute Force
2	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0	2
3	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.0	3
4	5.58 ± 1.28	5.61 ± 1.32	5.61 ± 1.32	6
5	6.46 ± 2.16	8.17 ± 2.89	7.76 ± 2.51	10
6	11.89 ± 3.94	16.1 ± 4.38	14.22 ± 4.54	20
7	19.26 ± 7.39	28.76 ± 7.97	26.06 ± 7.69	35
8	37.43 ± 12.61	55.13 ± 16.46	48.91 ± 15.04	70
9	68.35 ± 23.68	102.63 ± 23.38	98.44 ± 24.28	126
10	136.8 ± 52.15	194.1 ± 51.22	176.81 ± 51.94	252
11	295.72 ± 87.36	411.1 ± 63.22	388.68 ± 69.68	462
12	556.93 ± 165.3	759.63 ± 169.39	708.55 ± 164.4	924
13	1205.2 ± 281.42	1584.18 ± 138.62	1509.19 ± 180.79	1716
14	2297.13 ± 580.68	3063.99 ± 420.93	2911.47 ± 433.77	3432
15	4843.76 ± 1163.11	6128.21 ± 464.24	5886.66 ± 655.98	6435
16	9585.38 ± 1916.77	11941.86 ± 983.13	11670.18 ± 1032.98	12870
17	20469.16 ± 2764.06	23670.23 ± 1129.73	23415.26 ± 1158.72	24310
18	40178.97 ± 5502.29	47031.25 ± 2063.14	46001.55 ± 2978.96	48620
19	80829.82 ± 8060.97	90483.4 ± 2989.92	89862.87 ± 2998.67	92378
20	159455.12 ± 19584.34	181218.54 ± 5033.59	177919.18 ± 9104.83	184756
21	324270.11 ± 24256.73	350214.17 ± 2934.33	348522.41 ± 6246.37	352716
22	635444.81 ± 60558.07	696989.44 ± 11297.48	692638.87 ± 16162.25	705432
23	1280748.79 ± 70752.77	1345083.4 ± 10870.74	1343279.97 ± 18035.36	1352078
24	2552116.85 ± 112630.21	2690573.89 ± 17601.61	2687007.96 ± 19471.51	2704156
25	5046113.96 ± 164480.26	5190718.93 ± 13522.75	5186572.54 ± 31462.2	5200300

Mean number of leaf nodes explored ± 1 standard deviation over 100 solves

4.3 Application: Hydrophone Placement

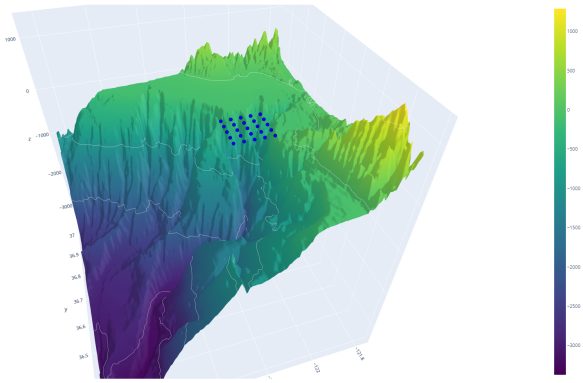
In this section, we consider the application of our contributions to sensor placement in the context of passive sonar. In this example, we seek to monitor the underwater acoustic environment in a region by placing a number of sonobuoys, each of which contains a single

omnidirectional hydrophone to receive acoustic signal from nearby vessel traffic. Using a discrete grid of possible sensor locations, we would like to place k sensors such that we maximize the mutual information between sensed and unsensed locations.

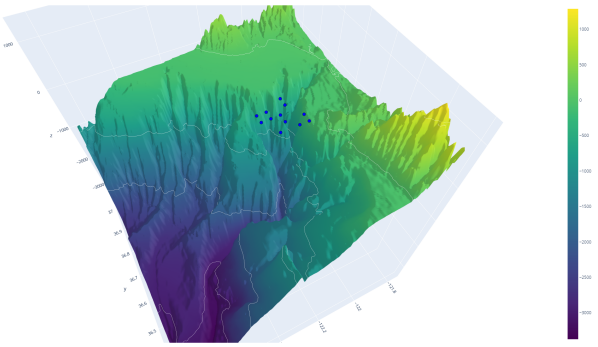
For this experiment, the region we seek to monitor is the Monterey Bay, off the coast of California. Figure 4.4 gives the bathymetric profile of the region and the candidate sensor locations considered.

Figure 4.4. Sensor Placements for Monitoring Vessel Traffic in the Monterey Bay

(a) The bathymetric profile and candidate sonobuoy locations for this experiment



(b) The sensor placement which maximizes mutual information using twelve sensors



Bathymetric data provided by NOAA [18]

We constructed the covariance matrix Σ of the candidate sensor locations as follows. Using historical AIS data of vessel traffic in the Monterey Bay throughout the year 2019[3], we sampled 1728 vessel locations and simulated acoustic propagation through the water column, measuring the strength of the signal (in dB relative to the emitted signal strength) received at each sensor. We assume that each vessel emits acoustic energy at the same amplitude. We conducted these acoustic simulations using the Bellhop ray tracing model provided through the implementation in the Acoustic Toolbox [5].

We note that this simulation approach incorporates a number of important contextual aspects of this problem. First, the ray tracing model incorporates the bathymetric features of the environment, which is especially critical to acoustic propagation in shallow water environments [20]. Additionally, the simulation allows acoustic experts to increase the accuracy of the simulation by incorporating additional information as it is available. For example, we use an idealized sound speed profile due to Munk [21] for our simulations due to lack of available sound speed profile data in our region of interest. If measurements of the sound speed profile in the Monterey Bay became available, one could easily incorporate it into the simulations before proceeding to determine the optimal sensor placement. In this way, the simulation used to estimate the covariance matrix of the candidate sensors is modular, and can be improved or modified as context and additional information permit without affecting the utility of our contributions. This is a distinct advantage of our approach over methods which rely on detection window assumptions, because in those methods assumptions on the received signal and the procedure to determine the optimal sensor placement are inseparably linked. We emphasize that this problem is more complex than one might assume by examining Figure 4.5.

[3]Provided by the US Coast Guard [19]

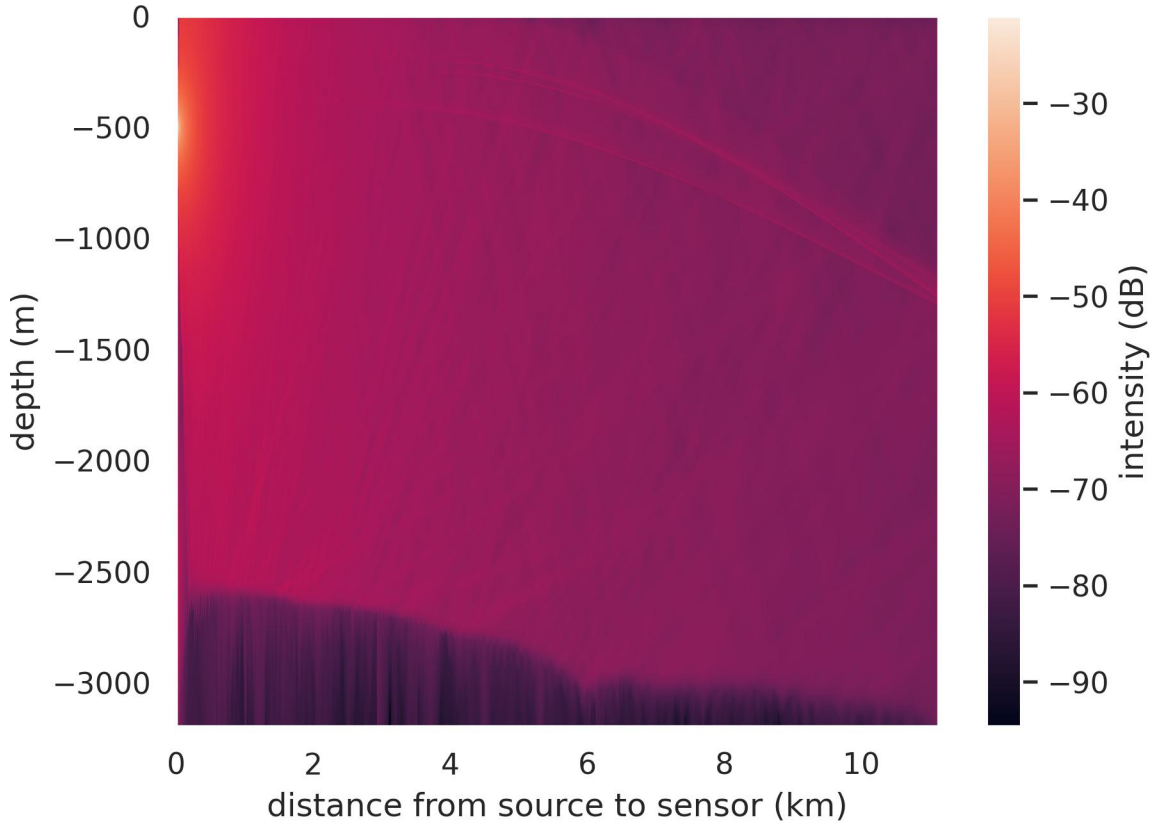


Figure 4.5. Acoustic Simulation Ray Tracing

In the figure, a few distinct paths are displayed where sound would propagate particularly far and could be detected at a considerable distance from the source. Additionally, the small regions of high intensity signals indicate that assuming a fixed radius of detection is inappropriate. The simulations we perform lend us the more complex structure of acoustic propagation. After conducting these simulations, we are left with a data matrix which contains a received signal intensity for each vessel/sensor pair. We then took the sample covariance matrix of the candidate sensor locations, and used this as the covariance matrix Σ in our branch and bound implementation, where we seek to optimally place 12 sensors among 25 candidate locations. The results of this experiment are given in figure 4.4. Solving this problem required 5227 SDP solves and cut 5,105,520 leaf nodes from the tree, with the result that we searched a total of 94,780 leaf nodes instead of the 5,200,300 required by brute force. We note that there are sensors placed in some of the shallow areas of the bay

closer to the eastern coast as well as towards Santa Cruz to the north. Additionally, there are sensors placed to be able to detect sound waves that would end up in the canyon in the middle of the graphic. This demonstrates the ability to input data about a region where we want to place sensors on a budget and have the algorithm output an optimal placement.

4.4 Conclusion

This thesis has presented a novel semidefinite relaxation to the problem of maximizing mutual information for placing acoustic sensors. In Chapter 1, we outline the importance of research in the area of spatial sensor placement and discuss the military relevance for littoral operations. Chapter 2 introduces the concept of mutual information as a criterion for building the maximization problem. Additionally, we introduce the theory behind and the semidefinite relaxation of our maximization problem. The branch and bound algorithm presented in Chapter 3 iteratively solves the SDPs in order to find the optimal placement of sensors given information in a covariance matrix. Finally, in the current chapter we present experiments and an application of the branch and bound algorithm to analyze effects of tuning parameters. We showed that when specifying a low depth when running the branch and bound algorithm primarily favors brute force when matrices are large. On the other hand, having full depth specified for placing sensors shows us that the feasible region is reduced by ~90% compared to brute force.

With the increasing need for remote sensing in littoral regions, we hope this thesis encourages others to continue developing our research. Further work on this topic could include finding better heuristics for generating bounds, as the outer rotation method for rounding did not seem to improve the performance of the branch and bound algorithm. This thesis mainly focused on acoustic sensors in the Monterey Bay, however, the SDP relaxation and branch and bound algorithm can apply to any spatial phenomena such as rainfall or temperature [9].

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: Proofs and Definitions

Proof of Proposition 1. Denote by $H(X)$ the differential entropy of a continuous random vector X, $H(X,Y)$ the joint information of two continuous random variables X and Y, and $I(X,Y)$ the mutual information between X and Y. Recall from [22, Section 8.5] that mutual information can be written in the equivalent form

$$I(X_A, X_{[N]\setminus A}) = H(X_A) + H(X_{[N]\setminus A}) - H(X_A, X_{[N]\setminus A}). \tag{A.1}$$

Finally, recalling from [22, Theorem 8.4.1] that the entropy of a Gaussian random vector $X \sim N(\mu, \Sigma)$ of length N is

$$H(X) = \frac{1}{2}\log\left[(2\pi e)^N \det(\Sigma)\right],$$

(A.1) becomes

$$\frac{1}{2}\log\left[(2\pi e)^{|A|}\det(\Sigma_A)\right] + \frac{1}{2}\log\left[(2\pi e)^{N-|A|}\det(\Sigma_{[N]\setminus A})\right] - \frac{1}{2}\log\left[(2\pi e)^N \det(\Sigma)\right] \tag{A.2}$$

$$= \frac{1}{2}\left[\operatorname{logdet}\Sigma_A + \operatorname{logdet}\Sigma_{[N]\setminus A} - \operatorname{logdet}\Sigma\right] \tag{A.3}$$

□

Proposition 2 *Let $X \sim N(\mu, \Sigma)$ be an n-dimensional random vector, and denote by $h(\cdot)$ the differential entropy of a random vector. The problem of selecting a subvector of X of size k to minimize differential entropy*

$$\min_{\max_{A\subseteq[n],|A|=k}} h(X_A)$$

is, up to constant factors, equivalent to maximizing the following

$$\max_{A\subseteq[n],|A|=k} \operatorname{logdet}(\Sigma_A). \tag{A.4}$$

Proof. For an n-dimensional Gaussian random vector with covariance Σ, its differential entropy is $\frac{1}{2}\left(n\log(2\pi e) + \operatorname{logdet}\Sigma\right)$ [22]. It follows that the differential entropy of $X_A|X_{[n]\setminus A}$ is[4]

$$\frac{1}{2}\left((N-k)\log(2\pi e) + \operatorname{logdet}\left(\Sigma_{[n]\setminus A} - \Sigma_{[n]\setminus A,A}\Sigma_A^{-1}\Sigma_{A,[n]\setminus A}\right)\right). \tag{A.5}$$

From the Schur complement [23],

$$\operatorname{logdet}\left(\Sigma_{[n]\setminus A} - \Sigma_{[n]\setminus A,A}\Sigma_A^{-1}\Sigma_{A,[n]\setminus A}\right) = \operatorname{logdet}\Sigma - \operatorname{logdet}\Sigma_A. \tag{A.6}$$

Substituting (A.6) into (A.5) and neglecting constant factors, we see that minimizing the differential entropy of $X_{[n]\setminus A}|X_A$ results in the optimization problem in (A.4).

□

[4]Here we have used the well-known result on the covariance of a conditional Gaussian

$$\operatorname{Var}(X_{[n]\setminus A}|X_A) = \Sigma_{[n]\setminus A} - \Sigma_{[n]\setminus A,A}\Sigma_A^{-1}\Sigma_{A,[n]\setminus A}.$$

List of References

[1] *Chief of Naval Operations Navigation Plan 2022*, NAVADMIN, Washington, DC, USA, 2022 [Online]. Available: https://media.defense.gov/2022/Jul/26/2003042389/-1/-1/1/NAVIGATION%20PLAN%202022_SIGNED.PDF

[2] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and vlsi," *Journal of the ACM (JACM)*, vol. 32, no. 1, pp. 130–136, 1985.

[3] H. González-Banos, "A randomized art-gallery algorithm for sensor placement," in *Proceedings of the seventeenth annual symposium on Computational geometry*, 2001, pp. 232–240.

[4] M. T. Kuhn, "Optimal sensor placement in active multistatic sonar networks," M.S. thesis, Dept. of Operations Research, NPS, Monterey, CA, USA, 2014 [Online]. Available: https://calhoun.nps.edu/handle/10945/42665

[5] M. B. Porter, *Acoustic Toolbox*, Oceanic Acoustic Library, 2022 [Online]. Available: https://oalib-acoustics.org/models-and-software/acoustics-toolbox/

[6] "Notes on cholesky factorization," class notes, Dept. of Computer Science University of Texas at Austin, Austin, TX, USA, March 2011.

[7] F. Pukelsheim, *Optimal Design of Experiments*, 2nd ed. Philadelphia, PA, USA: SIAM, 2006.

[8] W. Caselton and J. Zidek, "Optimal monitoring network designs," *Statistics Probability Letters*, vol. 2, no. 4, pp. 223–227, 1984 [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167715284900208

[9] A. S. Andreas Krause and C. Guestrin, "Near-optimal sensor placements in gaussian processes," *Journal of Machine Learning Research*, vol. 9, no. 8, Feb. 2008 [Online]. jmlr: https://jmlr.org/papers/volume9/krause08a/krause08a.pdf.

[10] D. C. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons, 2017.

[11] B. Gartner and J. Matousek, *Approximation Algorithms and Semidefinite Programming*, 1st ed. Berlin, DE: Springer, 2012.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*, 7th ed. Cambridge,UK: Cambridge University Press, 2009.

[13] U. Zwick, "Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems," 1999.

[14] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, vol. 169, no. 3, pp. 1042–1068, June 2016. Available: http://stanford.edu/~boyd/papers/scs.html

[15] R. Bassett and E. Vargas, "Sensor placements which maximize mutual information," GitHub, Aug. 21, 2022 [Online]. Available: https://github.com/rbassett3/mutual_info_sensor_placement

[16] S. Ghosh and S. G. Henderson, "Behavior of the NORTA method for correlated random vector generation as the dimension increases," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, no. 3, pp. 276–294, 2003.

[17] A. M. Law, "Simulation modeling and analysis," 2015.

[18] NNOAA NCEI Database. NOAA National Centers for Environmental Information. [Online]. Available: https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.nodc:0001255. Accessed Mar. 23, 2021.

[19] US Department of Homeland Security. USCG Navigation Center. [Online]. Available: https://www.navcen.uscg.gov/contact/ais-historical-request

[20] B. Katsnelson, V. Petnikov, and J. Lynch, *Fundamentals of shallow water acoustics*. Springer, 2012, vol. 1.

[21] M. B. Porter, *A Deep Water Problem: the Munk Profile*, Oceanic Acoustic Library, 1997 [Online]. Available: https://oalib-acoustics.org/website_resources/AcousticsToolbox/manual/node8.html

[22] T. M. Cover and J. A. Thomas, "Elements of information theory."

[23] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 2013. Available: https://books.google.com/books?id=5I5AYeeh0JUC

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California