2014-06

Optimal sensor placement in active multistatic sonar networks

Kuhn, Tobias Uwe

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMAL SENSOR PLACEMENT IN ACTIVE MULTISTATIC SONAR NETWORKS

by

Tobias Uwe Kuhn

June 2014

Thesis Co-Advisors:	Emily M. Craparo
	Craig W. Rasmussen
Second Reader:	Mümtaz Karataş

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	06-20-2014	Master's Thesis 11-06-2013 to 06-20-2014

4. TITLE AND SUBTITLE

OPTIMAL SENSOR PLACEMENT IN ACTIVE MULTISTATIC SONAR NETWORKS

5. FUNDING NUMBERS

6. AUTHOR(S)

Tobias Uwe Kuhn

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

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 document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Recently the idea of deploying non-collocated sources and receivers in multistatic sensor networks (MSNs) has emerged as a promising area of opportunity in sonar systems. This thesis addresses point coverage sensing problems in MSNs, where a number of points of interest have to be monitored in order to protect them from hostile underwater assets. We consider discrete "cookie cutter" sensors as well as various diffuse sensor models. By showing that the convex hull spanned by the targets is guaranteed to contain optimal sensor positions, we are able to limit the solution space. Using a cookie cutter sensor model, we are able to exclude even more suboptimal solutions by determining range-of-the-day, source and receiver circles.

To address the nonconvex single-source placement problem, we develop the Divide Best Sector (DiBS) algorithm, which quickly provides an optimal source position assuming fixed receivers. Starting with a basic implementation of DiBS, we show how incorporating advanced sector splitting methods and termination conditions further improve the algorithm. We also discuss two ways to use DiBS to find multiple source positions by placing sensors iteratively or simultaneously. Finally, we conclude that DiBS is a fast and simple algorithm that supports a wide variety of sensor models, various termination conditions, and objective functions.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Multistatic Sensor Network, Point Coverage Sensing, Optimization, Divide Best Sector	73
	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

OPTIMAL SENSOR PLACEMENT IN ACTIVE MULTISTATIC SONAR NETWORKS

Tobias Uwe Kuhn
Major, German Army
M.S., University of the German Armed Forces Munich, 2005

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN OPERATIONS RESEARCH
AND MASTER OF SCIENCE IN APPLIED MATHEMATICS**

from the

**NAVAL POSTGRADUATE SCHOOL
June 2014**

Author: Tobias Uwe Kuhn

Approved by: Emily M. Craparo
Thesis Co-Advisor

Craig W. Rasmussen
Thesis Co-Advisor

Mümtaz Karataş
Second Reader

Robert Dell
Chair, Department of Operations Research

Carlos Borges
Chair, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Recently the idea of deploying non-collocated sources and receivers in multistatic sensor networks (MSNs) has emerged as a promising area of opportunity in sonar systems. This thesis addresses point coverage sensing problems in MSNs, where a number of points of interest have to be monitored in order to protect them from hostile underwater assets. We consider discrete "cookie cutter" sensors as well as various diffuse sensor models. By showing that the convex hull spanned by the targets is guaranteed to contain optimal sensor positions, we are able to limit the solution space. Using a cookie cutter sensor model, we are able to exclude even more suboptimal solutions by determining range-of-the-day, source and receiver circles.

To address the nonconvex single-source placement problem, we develop the Divide Best Sector (DiBS) algorithm, which quickly provides an optimal source position assuming fixed receivers. Starting with a basic implementation of DiBS, we show how incorporating advanced sector splitting methods and termination conditions further improve the algorithm. We also discuss two ways to use DiBS to find multiple source positions by placing sensors iteratively or simultaneously. Finally, we conclude that DiBS is a fast and simple algorithm that supports a wide variety of sensor models, various termination conditions, and objective functions.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AoR area of responsibility

ASW anti-submarine warfare

CSV comma-separated values

DiBS Divide Best Sector

DSTO Defence Science and Technology Organisation

GAMS General Algebraic Modeling System

INLP integer nonlinear program

LHS left-hand side

LP linear program

MSN multistatic sensor network

PoI point of interest

RC receiver circle

RDC range of the day circle

SC source circle

USN United States Navy

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Active sonar systems have long constituted an important sensing mechanism aboard submarines and ships in anti-submarine warfare (ASW). Recently, however, the idea of deploying non-collocated sources and receivers has emerged as a promising area of opportunity in sonar systems. A multistatic sensor network (MSN) consisting of a number of non-collocated sources and receivers carries a number of advantages such as more complicated countermeasure tactics, improved deployment opportunities as well as reduced costs and noise pollution. The complex geometry of MSNs, however, makes deployment decisions like number of sensors and their locations more challenging.

Out of three sensing problems studied in the literature (area search, barrier search, and point coverage) this thesis addresses point coverage problems assuming *cookie cutter* sensor models as well as diffuse sensor models. We further assume stationary targets and sensors in a two-dimensional Euclidean plane with homogeneous environment. Additionally we neglect the target's aspect dependence and assume independence for all probabilities.

We start by showing that for each sensor location outside the convex hull encasing all targets, there exists a position inside whose detection probability is at least as good. Based on this, we limit the search for optimal sensor locations to the convex hull of the targets.

Next, we introduce the notion of range of the day circles (RDCs) for cookie cutter sensor models, i.e., circles around targets whose radii are the range of the day. Utilizing common points, we establish the minimal set of clusters of RDCs, $\widetilde{G}$, whose cardinality represents a lower bound on the number of sensors required to cover all targets. Moreover, $\widetilde{G}$ lets us define an upper bound on the number of detected targets with a restricted number of sensors. Receiver circles (RCs) and source circles (SCs) provide additional means to narrow down possible sensor locations.

Up to this point, we can formulate integer nonlinear programs (INLPs) with reasonable run time only to find optimal locations for exactly one source and one receiver, assuming a cookie cutter sensor model. Hence we develop the Divide Best Sector (DiBS) algorithm to find a single optimal source position assuming fixed target and receiver positions regardless of the applied sensor model. The algorithm divides the area of possible source locations

into rectangular sectors and evaluates their upper bounds. At each iteration we divide the sector with the highest upper bound into smaller sector. Eventually, the algorithm reaches a termination condition such as maximum size of a sector's longest edge or maximum optimality gap.

We proceed to investigate and assess some details of DiBS starting with discovering that a small number of newly created sectors per iteration as well as fewer initial sectors greatly reduce the number of total sectors created and evaluated by the algorithm. This subsequently leads to better run times.

Another method to further improve the algorithm is rotating the plane such that the edges of the minimum-area rectangle encasing all targets are parallel to x- and y- axis. By doing so, we minimize the sectors' area outside the convex hull.

Furthermore, we observe that uneven rectangles are able to increase the total number of sectors and therefore run time. We discuss two ways to address this issue by adjusting the termination condition and by creating only square sectors.

Finally, we explore two approaches to apply DiBS to problems that require finding an optimal placement for multiple sources. Here, the iterative method provides a lower and upper bound assuming a cookie cutter sensor model. The second approach places sources simultaneously and evaluates all possible combinations of sectors. This approach, however, can only be applied to small numbers of sources.

CHAPTER 1:
Introduction

1.1 Background

Active sonar systems have long constituted an important sensing mechanism aboard submarines and ships in anti-submarine warfare (ASW). In a typical sonar system, a ping is sent out and the echo yields information about other objects in the area. Recently, however, the idea of deploying non-collocated sources and receivers has emerged as a promising area of opportunity in sonar systems.

A multistatic sensor network (MSN) consisting of a number of non-collocated sources and receivers carries a number of advantages. Cox (1989, p. 23) states that "countermeasure tactics are greatly complicated if the target does not know the position of the receivers." This is justified by the fact that receivers do not send out pings and thus do not reveal their locations. Beyond that, procurement estimates of the United States Navy (USN) indicate that sources might cost about five times as much as receivers (USN, 2014). Thus, deploying more receivers than sources might significantly reduce costs without sacrificing performance. Also, a "multistatic system can employ different platforms for sources and receivers. A ship might be the source, while the receivers are sonobuoys" (Washburn, 2010, p. 1). On top of this, a ping might be received by multiple receivers. Coon (1997) and Simakov (2008) discuss how to merge multiple detections into a single alert that is more precise and eliminates some of the false alarms that occur on traditional (monostatic) sonar systems.

The performance of a bistatic sonar system, i.e., an MSN with exactly one source and one receiver, is significantly more difficult to model than a monostatic sonar system, leading to challenges in optimal deployment and usage. The primary source of these challenges lies in the differences of the geometry of both systems. In a monostatic sonar system the detection probability is mainly related to the distance between the sonar device and the potential target. This relationship is more complicated in a bistatic model, where the detection probability depends on the product of the target-source and target-receiver distances (Cox,

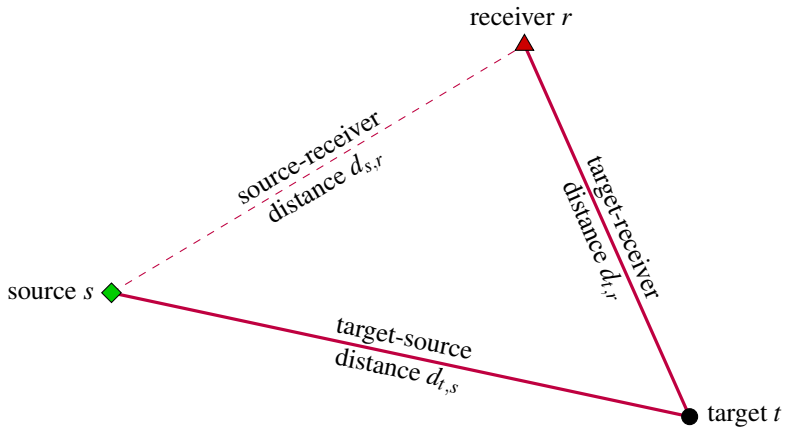

Figure 1.1: **Geometry of a Bistatic Sonar System** - The detection probability for target t depends on the product of the target-source and target-receiver distance.

1989) as displayed in Figure 1.1. Moreover, the analytical challenges are exacerbated by MSNs involving multiple sources and receivers.

1.2 Literature Review

Researching sonar technology and MSNs in particular is a wide field with most diverse subcategories. We will discuss a selection in this section, starting with the geometry of multistatic models and following with examples for usage and deployment.

1.2.1 The Geometry of Multistatic Sensor Networks

Multiple authors describe the geometry of MSNs. Cox (1989) analyzes the relationship between monostatic and bistatic active sonars. He derives that a detection probability contour, i.e., a contour consisting of all locations for a target t with the same detection probability, is defined by the constant product

$$d_{t,s} \times d_{t,r} = \rho_{t,s,r}^2, \tag{1.1}$$

where $\rho_{t,s,r}$ is constant, and $d_{t,s}$ and $d_{t,r}$ are the distances from a target t to a source s and to a receiver r respectively. The distances are also illustrated in Figure 1.1. Those contours are geometric figures known as Cassini ovals shown in Figure 1.2. The team of Matthew Fewell and Sylvia Ozols from the Australian Defence Science and Technology

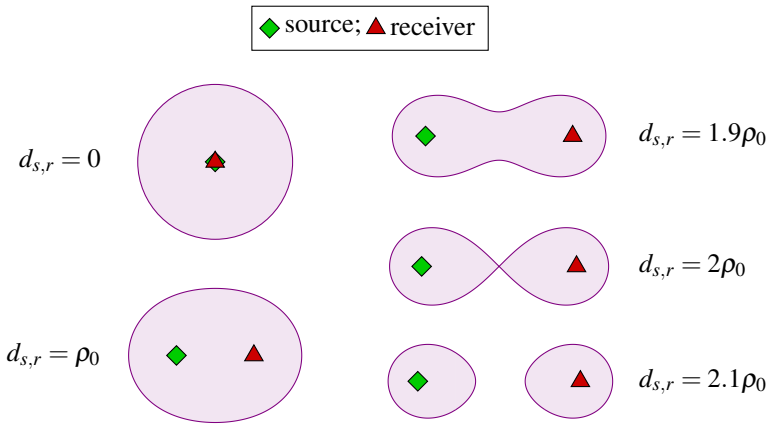

Figure 1.2: **Examples of Cassini Ovals** - Cassini ovals are shown with respect to source and receiver distance $d_{s,r}$ where ρ_0 denotes the range of the day. If source and receiver are collocated then the Cassini oval is a circle that corresponds to a monostatic sensor. Increasing the distance changes the shape from oval over dog-bone shape up to two separate egg shapes.

Organisation (DSTO) published multiple reports on MSNs. Fewell and Ozols (2011, p. 4) state that the constant $\rho_{t,s,r}$ from Equation (1.1) is the equivalent range to the range from a monostatic sonar system, which enables us to compute detection probabilities for bistatic sensor networks. The authors point out some issues with this model, e.g., "that not only [the detection probability is] high at a receiver but also the value is unaffected by the source-receiver distance. This seems inadequate (Fewell & Ozols, 2011, p. 7)." They list various possibilities to modify the model to account for these issues.

There are multiple ways to determine $P_{t,s,r}$, the probability to detect target t with source s and receiver r in a multistatic model as displayed in Figure 1.3. The simplest one is the *cookie cutter* sensor model, also known as the *definite range* sensor in literature. Based on the *range of the day* ρ_0, i.e., the distance from a monostatic sensor to a target where the detection probability is 50%, we define $P_{t,s,r}$ as

$$P_{t,s,r} = \begin{cases} 1 & \text{if } \rho_b \leq \rho_{t,s,r} \leq \rho_0, \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

Other names for the range of the day are the *detection range* or R_{50}. The *blind zone* ρ_b is the area where targets cannot be detected because their echoes arrive at nearly the same time as the ping from the source. Then the total detection probability for target t, P_t, i.e.,

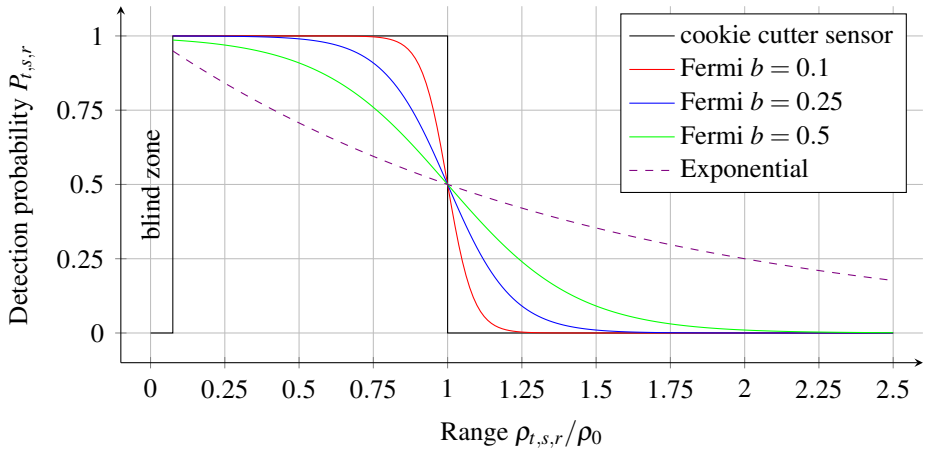

Figure 1.3: **Sensor Models** - The probability curves for the three sensor models cookie cutter, Fermi and exponential function are displayed. Range is expressed as multiples of the range of the day ρ_0. All models have a $P_{t,s,r} = 0.5$ at $\rho_{t,s,r}/\rho_0 = 1$ to be consistent with the definition of the range of the day.

the probability to detect target t with all sensors in scene, is defined as

$$P_t = \max_{(s,r)\in S\times R} P_{t,s,r}. \tag{1.3}$$

This means, if at least one pair of sensors detects target t then $P_t = 1$, otherwise $P_t = 0$. Although Equations (1.2) and (1.3) form an analytically convenient model, it lacks some features of real sensors such as a gradually decreasing $P_{t,s,r}$ with increasing ρ.

Hence, the DSTO team proposes two diffuse sensor models in (Fewell & Ozols, 2011). The first is the *Fermi* function[1]

$$P_{t,s,r} = \begin{cases} \dfrac{1}{1+10^{(\rho_{t,s,r}/\rho_0-1)/b}} & \text{if } \rho_{t,s,r} \geq \rho_b, \\ 0 & \text{otherwise.} \end{cases} \tag{1.4}$$

The diffusivity parameter b determines how rapidly probability values change when changing range ρ. As $b \to 0$, the Fermi function approaches the cookie cutter model.

[1]Discovered in 1926 by Enrico Fermi and Paul Dirac when researching electron behavior.

The second model is the exponential function

$$P_{t,s,r} = \begin{cases} 10^{-0.30103\rho_{t,s,r}/\rho_0} & \text{if } \rho_{t,s,r} \geq \rho_b, \\ 0 & \text{otherwise.} \end{cases} \tag{1.5}$$

The exponent is chosen such that if $\rho_{t,s,r}/\rho_0 = 1$ the detection probability $P_{t,s,r} = 0.5$. This is important to be consistent with the definition of the range of the day.

For diffuse sensor models we need a different approach to evaluate the total detection probability. Thus, we define a more general version of Equation (1.3) to compute P_t regardless of the chosen sensor model assuming all probabilities are independent as

$$P_t = 1 - \prod_{(s,r)\in S\times R} (1 - P_{t,s,r}). \tag{1.6}$$

In order to account for different weighted targets, we introduce v_t as the value of target t. Hence we denote the expected reward for target t as $v_t P_t$. If all targets have the same value, we simplify the expected reward for target t by using P_t. Based on this, we can define multiple objective functions depending on the chosen sensor model. While maximizing the total expected reward is our only objective when using a cookie cutter model, we can also choose to maximize the minimum expected reward. The objective functions are summarized in Table 1.1.

Objective	Sensor Model	Formula
maximize total expected reward	cookie cutter	$\max \sum_{t\in T} v_t P_t$
maximize average expected reward	diffuse	$\max \sum_{t\in T} v_t P_t / \lvert T \rvert$
maximize minimum expected reward	diffuse	$\max \min_{t\in T} v_t P_t$

Table 1.1: **Objective functions** - Three different objective functions are possible depending on the chosen sensor model, where v_t denotes the value or weight of target t.

1.2.2 Deployment and Usage

There are different approaches to quantify the effectiveness of the deployment and usage of multistatic sensors. A strategic rather than a tactical approach is analyzed by Washburn (2010). He assumes that sources and receivers are deployed uniformly at random within

some region. The author develops methods for approximating the detection probabilities as a function of number of sensors. The most interesting aspect about this approach is the fact that it does not need to consider the geometric arrangement of the sensors.

The DSTO analyzes multiple scenarios. Fewell and Ozols (2011) compare a field of monostatic sonar systems with that of a field of similar sonars operated multistatically, where sources and receivers are collocated. The direct comparison of the two modes of operation reveals that the correct choice of sensor models affects the outcome. Using a cookie cutter sensor model, the researchers find no advantage to the MSN. However, using the exponential model, the researchers achieved the same results in an MSN with about one quarter the number of sensors as the monostatic sensors. This finding is important since it reduces the number of pings in a given field. As soon as a ping is sent out, a hostile submarine knows the location of the source and will depart, which consequently makes it harder to detect. Another reason to reduce the number of pings is the artificial stress for sea dwellers produced by sonar systems.

There are three main types of sensing coverage problems in the literature: area search, barrier search, and point coverage. Using the same sensor models as in (Fewell & Ozols, 2011), Ozols and Fewell (2011) compare various layouts for sensor positions in a MSN, where a large area has to be covered. From the 27 layouts tested, four are recommended for use depending on the ratio of source and receiver cost. Additionally, in a classified report, Fewell, Ozols, and Rzetelski (2011) discuss the deployment of MSNs for a barrier search problem. Gong, Zhang, Cochran, and Xing (2013) address the barrier search problem by finding the optimal deployment of multistatic sensors to maximize the worst-case intrusion detectability. The authors show that a balanced structure yields the optimal solution.

Craparo and Karataş (2014) engage the point coverage problem for MSNs. The idea behind is to monitor a number of points of interest (PoIs), e.g., oil platforms, aircraft carriers, ports, etc., in order to detect hostile underwater assets. The authors assume fixed PoIs and receivers, and discuss various approaches to optimally place multiple sources. To our knowledge this is the only existing study considering the point coverage problem for MSNs.

1.3 Objectives

From the three main types of sensing coverage problems (barrier search, area search and point coverage), this thesis focuses on the point coverage problem for MSNs. The overall question is: where do we place sources and receivers such that the PoIs are optimally covered? To be in accordance with the terminology of other literature, 'PoIs' are called 'targets', and instead of 'covering' we say 'detecting'. To engage this problem, we consider the following questions.

- What are basic observations regarding point coverage sensing in MSNs? Particularly,
  - Is the convex hull spanned by the targets guaranteed to contain the optimal locations of sources and receivers?
  - Are there other ways to exclude suboptimal solutions?
- How can we find an optimal placement in MSNs? Particularly,
  - Are there models or algorithms that represent point coverage sensing?
  - How do the developed means perform and how can they be improved?

1.4 Scope, Limitations, and Assumptions

For the developed models we assume that all targets and sensors are stationary and exist in a two-dimensional Euclidean plane with homogeneous environmental conditions. In the course of this study, all three detection probabilities shown in Figure 1.3 are considered. When we build our models, however, we start with the simple cookie cutter model and later advance to the diffuse sensor models. The effect of the blind zone can be greatly reduced by pulse compression as shown by Fewell and Ozols (2011, p. 11). For this reason, we do not consider blind zones in this thesis, i.e., we assume $\rho_b = 0$. Furthermore, we neglect the target's aspect dependence and assume all probabilities are independent. A set of notations used throughout the study is summarized in Table 1.2.

1.5 Contributions And Outline

In the first part of this thesis we explore some general observations about point coverage sensing with MSNs. We prove that a convex hull encasing the targets contains optimal sensor locations. Furthermore, we show the importance of various circles and the *clusters* of targets induced by their intersections for cookie cutter sensor models. We show that these

Element	Set	Description
t	T	target
s	S	source
r	R	receiver
v_t	$\mathbb{R}^+$	value of target t
(x_i, y_i)	$\mathbb{R}^2$	coordinates of object i, $\forall i \in T \cup S \cup R$
$d_{i,j}$	$\mathbb{R}^+$	Euclidean distance between object i and j, $\forall i, j \in T \cup S \cup R$
ρ_0	$\mathbb{R}^+$	range of the day
$\rho_{t,s,r}$	$\mathbb{R}^+$	equivalent monostatic range for target t, source s, and receiver r
$P_{t,s,r}$	$\mathbb{R}^+$	probability to detect target t with source s and receiver r; $P_{t,s,r} \leq 1$
P_t	$\mathbb{R}^+$	total detection probability for target t; $P_t \leq 1$

Table 1.2: **Notation** - This table summarizes the notation used throughout the study.

clusters can be used to define bounds on the sensing problem. Additionally, we discuss how integer nonlinear programs (INLPs) can be used to find an optimal solution.

The second part focuses on the development and enhancement of a new algorithm to find the optimal source position, assuming fixed targets and receivers in place. We investigate many details of the algorithm and assess methods and means to improve it.

CHAPTER 2:
Observations On Point Coverage Sensing

This chapter analyzes the point coverage scenario and discovers some basic observations. Those observations help to limit the solution space and exclude infeasible or non-optimal solutions. We also show how to apply the gained insights and whether the outcome justifies utilizing the developed method.

2.1 The Convex Hull

A simple approach to limit the set of potential sensor locations would be defining a rectangle that contains all PoIs. Even though this can easily be implemented, we want to exclude as many locations as possible. A more sophisticated approach involves looking at the convex hull spanned by the targets. As vividly described in (De Berg, Van Kreveld, Overmars, & Schwarzkopf, 2000, p. 3), one can imagine the targets as nails sticking out of the plane. If we hold an elastic rubber band around the nails and let it go, the enclosed area will be the convex hull.

2.1.1 Properties Of The Convex Hull

The example in Figure 2.1 shows a convex hull for a particular set of targets. It also shows that all targets are either inside the hull or at one corner of the resulting polygon. The latter are called vertices. Hence we define the set C as

$$C = \{t \in T | t \text{ is a vertex of the convex hull of } T\}.$$

The convex hull is defined as the set $\text{Conv}(T)$, that contains all points, that are spanned by the vertices in T. It is easy to see that $\text{Conv}(T) = \text{Conv}(C)$. If all nails that are not vertices are removed in the previously mentioned example, the rubber band would still enclose the same area. Alongside this picturesque description of a convex hull, there also exists a mathematical definition. Each point $p \in \text{Conv}(C)$ with coordinates (x_p, y_p) can be written as a linear combination of the vertices in C. This is shown in Equations (2.1), where $0 \leq \lambda_t \leq 1$ for all $t \in C$.

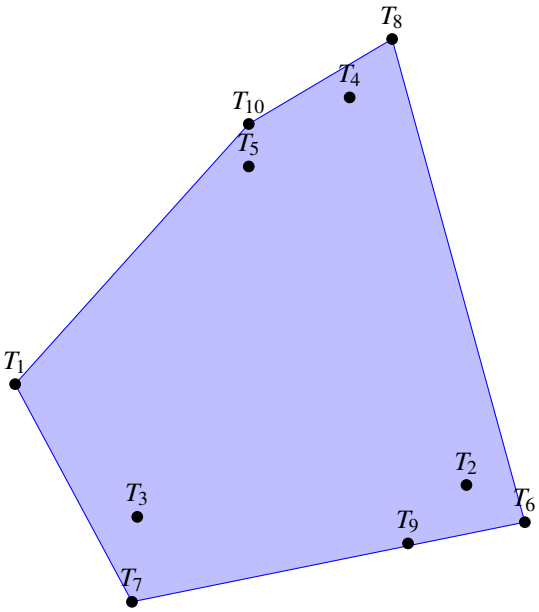

Figure 2.1: **Convex Hull Example** - A Convex Hull spanned by a set of targets. Its vertices are T_7, T_6, T_8, T_{10} and T_1.

$$\begin{aligned} x_p &= \sum_{t \in C} \lambda_t x_t, \\ y_p &= \sum_{t \in C} \lambda_t y_t, \\ 1 &= \sum_{t \in C} \lambda_t. \end{aligned} \tag{2.1}$$

Theorem 2.1 describes the relationship between the convex hull and the detection probability for sensor p, where $P_t(p)$ denotes $P_{t,p,r}$ if p is a source and $P_{t,s,p}$ otherwise.

Theorem 2.1. *For every sensor position $p \notin \mathrm{Conv}(C)$, there exists a position $p' \in \mathrm{Conv}(C)$, such that $P_t(p) \leq P_t(p'), \forall t \in T$.*

Proof. Assume sensor position $p \notin \mathrm{Conv}(C)$. Let $p' \in \mathrm{Conv}(C)$ denote the position inside the convex hull with the shortest distance to p. Then p' is either on an edge of the convex hull or collocated with one of its vertices as shown in Figure 2.2. In both cases, the convex hull and with it all targets $t \in T$ are behind an imaginary line perpendicular to the line from p to p' crossing that line at p'. Otherwise there would be another p' that is closer to p, which is a contradiction.

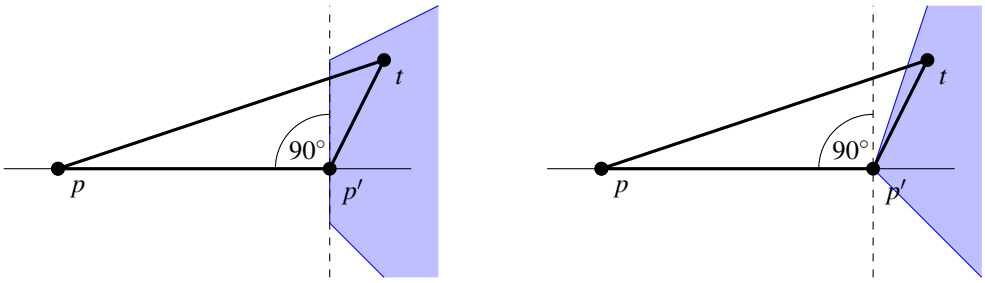

Figure 2.2: **Theorem 2.1 Proof** - There exist two cases for the closest position $p' \in \text{Conv}(C)$ to p. The left figure displays p' on an edge of the convex hull, while p' is collocated with a vertex on the right. In both cases all targets $t \in T$ are right of the dashed line, which is perpendicular to the line from p to p' and crosses p'.

Let t be a target inside the convex hull. Without loss of generality, we assume that p and $p'$are lying on a line parallel to the x-axis. Then the horizontal distance between p and t is $|x_p - x_t| = |x_p - x_{p'}| + |x_{p'} - x_t|$ and we derive

$$\begin{aligned} d_{t,p}^2 &= (x_p - x_t)^2 + (y_p - y_t)^2 \\ &= (|x_p - x_{p'}| + |x_{p'} - x_t|)^2 + (y_p - y_t)^2 \\ &\geq (x_{p'} - x_t)^2 + (y_{p'} - y_t)^2 \\ &= d_{t,p'}^2. \end{aligned}$$

Let $\rho_t(p)$ denote $\rho_{t,p,r}$ and $\rho_{t,s,p}$ if p is a source or receiver respectively. Thus the equivalent monostatic range for p' is

$$\rho_t(p') = \sqrt{d_{t,p'} d_{t,x}} \leq \sqrt{d_{t,p} d_{t,x}} = \rho_t(p),$$

where x is the second type of sensor needed in an MSN. It follows that the detection probability for a target t using sensor p', $P_t(p') \leq P_t(p)$ since all sensor model functions are monotonically nonincreasing with distance as is shown in Figure 1.3. □

We conclude that applying Theorem 2.1 we are able to place all sources and receivers inside the convex hull without sacrificing detection probabilities. Hence, we are now able to constrain our solution space, i.e., we can find optimal positions for each sensor inside the convex hull. We now describe and assess two ways to construct the convex hull.

2.1.2 A Linear Program To Find Vertices

The first method uses a linear program (LP) to find the vertices of the convex hull. We use the fact that vertices themselves cannot be written as a linear combination of other points, whereas targets that are not vertices can. Based on this insight, we can define the following LP.

Indices and Sets:

$t, t' \in T$ targets.

Data [units]:

x_t x coordinate of target t [unitless],

y_t y coordinate of target t [unitless].

Decision Variables:

$\lambda_t^{t'}$ coefficient for target t's position subject to target t.

Formulation:

$$\min_{\boldsymbol{\lambda}} \sum_{t \in T} \lambda_t^t$$

Subject to

$$x_{t'} = \sum_{t \in T} \lambda_t^{t'} x_t \qquad \forall t' \in T, \tag{2.2a}$$

$$y_{t'} = \sum_{t \in T} \lambda_t^{t'} y_t \qquad \forall t' \in T, \tag{2.2b}$$

$$1 = \sum_{t \in T} \lambda_t^{t'} \qquad \forall t' \in T, \tag{2.2c}$$

$$0 \leq \lambda_t^{t'} \qquad \forall t, t' \in T.$$

The LP tries to minimize the number of target locations not expressed as a convex combination of other target locations. For a non-vertex target t λ_t^t can be zero, since t's position can be represented by a linear combination of the vertices. The only possible value for vertices, however, is one. Thus, we can define the set of vertices C as

$$C = \{t \in T | \lambda_t^t = 1\}.$$

Implementing and solving this LP in General Algebraic Modeling System (GAMS) runs very quickly for small numbers of targets. Figure 2.3, however, shows that more targets result in an disproportional increase in run times when using the simplex algorithm. In fact, Klee and Minty (1972) show that in a worst-case scenario, the simplex algorithm can reach exponential complexity.

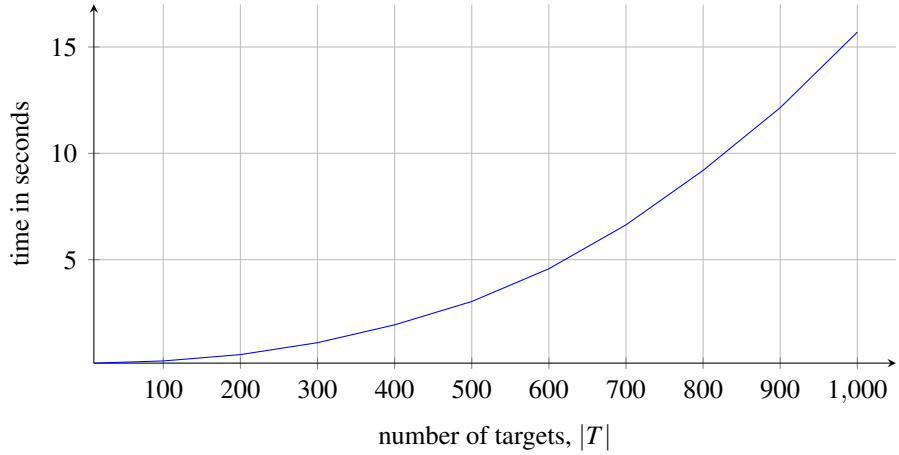

Figure 2.3: **Convex Hull LP Run Time** - For small numbers of targets the LP runs very quickly. Run Times, however, grow very quickly with the number of targets.

2.1.3 The Graham Scan Algorithm

Another algorithm to find the vertices of a convex hull with complexity $O(|T|\log|T|)$ is the Graham scan described in (Graham, 1972). The algorithm starts with finding $t_0 \in T$, the target with the smallest value for y_t. If there are multiple targets that share the smallest y_t, we pick the one with the smallest x_t out of the candidates.

Next we compute the angles θ_t each target $t \in T$ makes with t_0 and the x-axis using the formula

$$\theta_t = \text{atan}^2(y_t - y_{t_0}, x_t - x_{t_0}). \tag{2.3}$$

Sorting the targets by θ_t starting with the smallest, the algorithm now considers each target as possible vertex. At each step it is determined whether the current target t and its two

predecessors $t-1$ and $t-2$ make a turn clockwise or counter clockwise by calculating

$$c = (x_{t-1} - x_{t-2}) \times (y_t - y_{t-2}) - (y_{t-1} - y_{t-2}) \times (x_t - x_{t-2}). \tag{2.4}$$

If the result is positive the points make a counter clockwise turn and the next target in the ordered list is considered. If $c < 0$ then the turn is counter clockwise and the middle target $t-1$ is identified as non-vertex. At $c = 0$ all three targets are collinear, which also leads to discarding $t-1$. Figure 2.4 displays the working of the Graham scan algorithm.

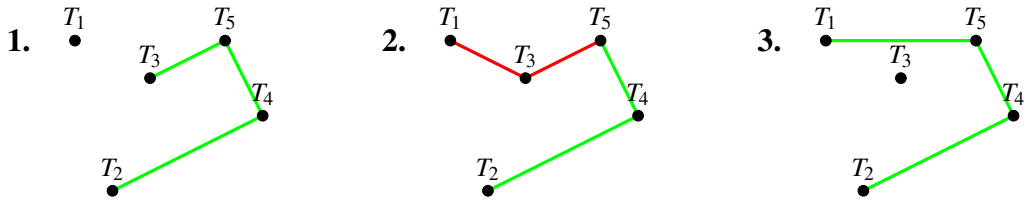

Figure 2.4: **Graham Scan** - The working of the Graham scan is displayed. Targets are considered in order of the angle they make between the lowest target and the x-axis. In this example at the second step the algorithm detects a clockwise turn, which leads to discarding the middle target on step 3.

The low complexity of the Graham scan is a result of the simple formula in Equation (2.4) to decide the complex issue about clockwise or counterclockwise turns. Outputting the vertices of the convex hull in a counter clockwise order is another advantage we later exploit in Section 3.4.3.

2.2 Range Of The Day Circles

This section analyzes the relationship between the range of the day, ρ_0 and the cookie cutter sensor model. We already know from Equation (1.2) that a target is detected if and only if its equivalent monostatic range $\rho_{t,s,r} \leq \rho_0$. Besides that, we introduce range of the day circles (RDCs), i.e., circles around targets with radius ρ_0 as shown in Figure 2.5. In the following, we discuss methods that use RDCs that bound the problem. Even though we develop these methods for cookie cutter sensor models, they also apply to problems where we have to meet a particular detection probability, e.g., 80%. Here we simply change the radius to a value that corresponds with the demanded probaility.

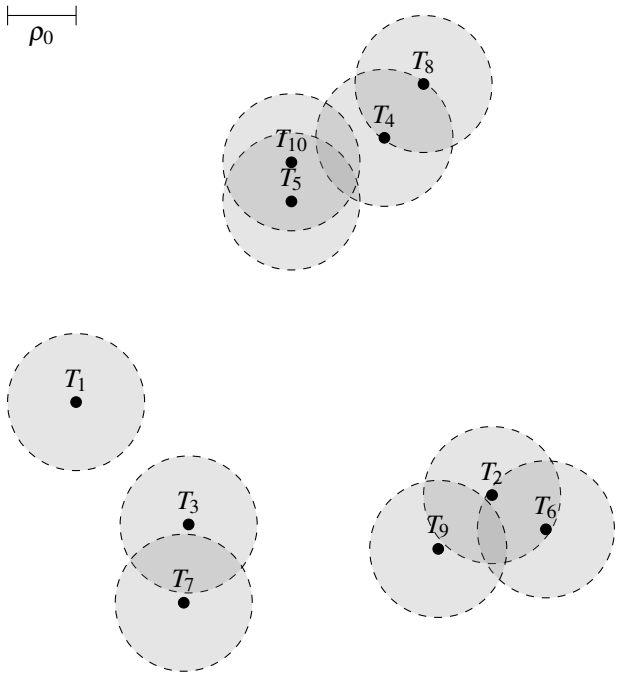

Figure 2.5: **Range Of The Day Circles Example** - Assuming a cookie cutter sensor model, range of the day circles determine bounds for the problem. Each circle has a target for center and ρ_0 for radius.

2.2.1 Properties

An important observation is the relationship between sensor positions relative to RDCs and a target's detection probability.

Theorem 2.2. *A target t is detected only if at least one sensor is inside t's RDC.*

Proof. To arrive at a contradiction, we assume all sensors are located outside the RDC of target t and $P_{t,s,r} = 1$ for some $s \in S$ and $r \in R$. Then $d_{t,s} > \rho_0, \forall s \in S$ and $d_{t,r} > \rho_0, \forall r \in R$. It follows that

$$\begin{aligned} \rho_{t,s,r} &= \sqrt{d_{t,s} \times d_{t,r}} \\ &> \sqrt{\rho_0 \times \rho_0} = \rho_0. \end{aligned}$$

But then according to Equation (1.2) $P_{t,s,r} = 0$, which is a contradiction. □

This, however, does not mean that we can find an optimal sensor placement by solely look-

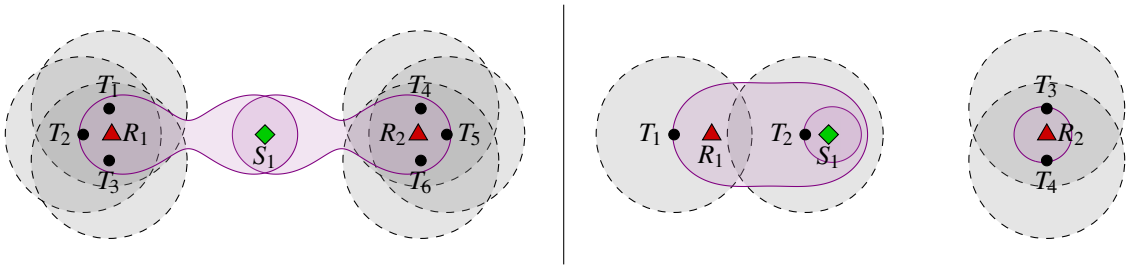

Figure 2.6: **Range Of The Day Circles Properties** - The left plot illustrates that an optimal placement can be found by placing the source S_1 outside RDCs. Forcing the S_1 inside the circles results in losing coverage of targets. The right plot shows an example where an optimal receiver position is outside RDC intersections. Here, it is not possible to cover all targets by placing all receivers into RDC intersections.

ing at positions inside RDCs. The left plot in Figure 2.6 demonstrates a counterexample where the optimal position for the source is located between both cliques of targets outside any RDCs. Moving this source inside a RDC of one of the cliques results in losing coverage of targets in the other clique.

A reasonable heuristic may involve choosing to place receivers into the circles and let the sources be the connecting elements of the MSN in order to keep the number of sources as small as possible. However, the impression that optimal receiver positions can only be found in RDC intersections, unless a circle does not have an intersection with another circle, is fallacious as pictured by the right plot in Figure 2.6. Receiver R_1 cannot be moved into the intersection of T_1's and T_2's RDCs without sacrificing coverage of T_1. Covering targets T_3 and T_4 inhibits the source from moving closer to T_1. We take a closer look at this example in Section 2.2.3.

2.2.2 Clusters

Nonetheless we can use RDCs and their intersections to form clusters. A cluster $G \subseteq T$ is a maximal set of targets, where the RDCs of all targets $t \in G$ have at least one point in common. Since we require a cluster to be maximal, it is not possible to add another target to a cluster. In our example from Figure 2.5 we can find the clusters

$$G_1 = \{T_1\}, \qquad G_2 = \{T_3, T_7\}, \qquad G_3 = \{T_2, T_6, T_9\},$$
$$G_4 = \{T_4, T_5, T_{10}\}, \qquad G_5 = \{T_4, T_8\}.$$

We make some observations about clusters. First of all, each target $t \in T$ occurs in at least one cluster. If there are no intersections with circles from other targets, a target forms its own cluster, e.g., T_1 is the only element of G_1. Additionally, it is possible that a target is a member of multiple clusters, e.g., $T_4 \in G_4$ and $T_4 \in G_5$. Moreover, it is important to notice that even though at a first glance it looks like the clique of targets on top of Figure 2.5 (T_4, T_5, T_8 and T_{10}) all belong together, they actually form two separate clusters since T_8 is only connected to T_4.

Figure 2.7 shows a special case: the left plot forms exactly one cluster, while the right plot forms three clusters $\{T_1, T_2\}$, $\{T_2, T_3\}$ and $\{T_1, T_3\}$ since the three targets do not have a single point in common, even though the targets' RDCs mutually intersect.

Figure 2.7: **Clusters Special Case** - The targets in the left plot form exactly one cluster. The targets on the right, however, form the three clusters $\{T_1, T_2\}$, $\{T_2, T_3\}$ and $\{T_1, T_3\}$ since the three targets do not have a single point in common.

Based on this, we define the minimal set of clusters $\widetilde{G}$ as the smallest set of clusters that contains all targets $t \in T$. For the right plot in Figure 2.7 a minimal set of clusters is $\widetilde{G} = \{\{T_1, T_2\}, \{T_2, T_3\}\}$. Furthermore, we formulate the following theorem.

Theorem 2.3. *A lower bound on the number of sensors required to detect all targets is* $|\widetilde{G}|$.

Proof. To arrive at a contradiction, we assume that all targets are covered with $|\widetilde{G}| - 1$ sensors. It follows from Theorem 2.2 that at least one sensor has to be inside each target's RDC. Since the circles of each cluster have at least one point in common, putting a sensor on that point results in having a sensor in each RDC of this cluster. Moreover, since clusters are maximal, we cannot add another target. Hence, one cluster and therefore at least one target remains without a sensor in its respective RDC. □

As a result from Theorem 2.3 we can define $|\widetilde{G}|$ as a lower bound for the number of sensors required to detect all targets. By the same token, we are able to find an upper bound for the number of targets we can cover with a given number of sensors n by choosing n sets from $\widetilde{G}$, such that the number of contained targets is maximized.

2.2.3 Other Circles

Next to RDCs, there exist other circles that provide insight into where to place either sources or receivers. Craparo and Karataş (2014) introduce the notion of the *detection disk* for each target t and receiver r that has t for center and its radius defined by

$$r = \frac{\rho_0^2}{d_{t,r}}.$$

In order to better distinguish detection disks from a second type of circles we introduce in this section, we refer to them as receiver circles (RCs). In order to detect a target t, we have to place a source inside t's RC.

Figure 2.8 shows the example from Figure 2.6 augmented with its RCs. Here the red and blue circles are RCs for R_1 and R_2, respectively. Using R_1 and one source we are able to detect T_1 and T_2 simultaneously since their RCs overlap. If we want to detect T_3 or T_4, however, we are not able to detect more than one target since their R_1 RCs do not overlap with other R_1 RCs. On the other hand, the R_2 RCs of T_3 and T_4 overlap and even have a common point with the RC of T_2, such that placing a source here results in detecting T_2, T_3 and T_4 with R_1.

By the same token, we can define source circles (SCs) whose radius is determined by a source location as

$$r = \frac{\rho_0^2}{d_{t,s}}.$$

Figure 2.9 displays the SCs in our previous example. With the shown fixed source position, placing a receiver at the intersection of T_3's and T_4's SCs ensures detecting both targets as well as T_2. Placing a second receiver somewhere into T_1's SC always detects T_2 also since T_1's circle is completely overlapped by T_2's SC.

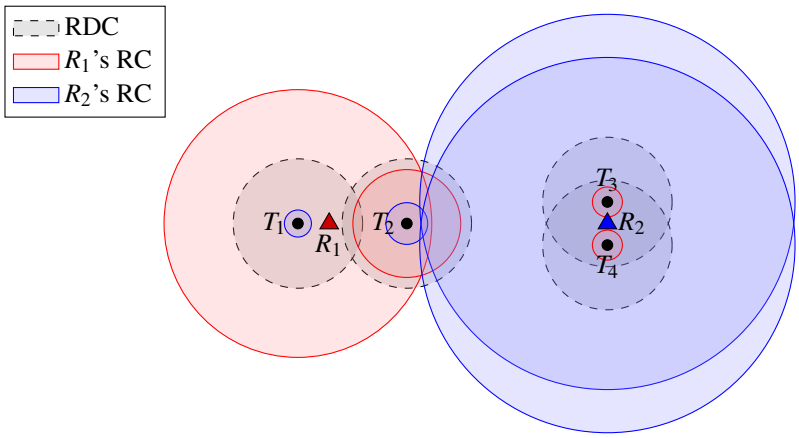

Figure 2.8: **Receiver Circles Example** - Red and blue circles represent receivers R_1's and R_2's receiver circles respectively. Placing a source at RC intersections ensures detection of respective targets.

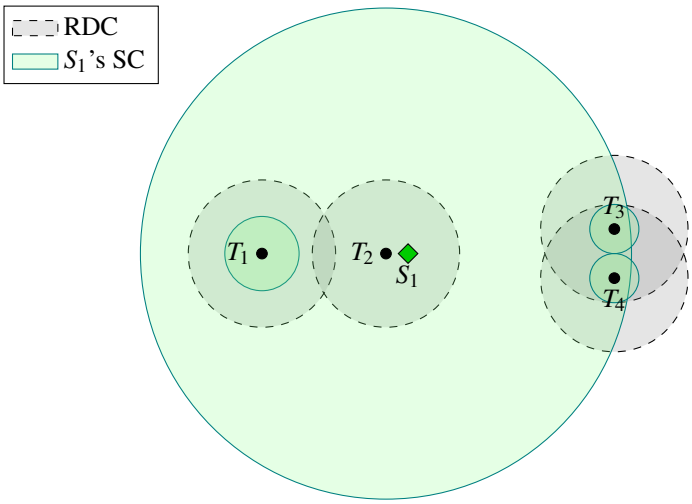

Figure 2.9: **Source Circles Example** - Placing a receiver at the intersection of targets T_3's and T_4's source circles guarantees detecting both targets. A receiver somewhere in T_1's SC always detects T_2 as well because of the total overlapping.

Assuming the positions of one type of sensors is already fixed, utilizing RCs and SCs is a way to decide positions for the remaining type. Nevertheless, a large number of targets and sensors makes their manual use hardly practical.

2.3 Utilizing Nonlinear Programs

We now investigate the usage of INLPs to find optimal sensor placements in MSNs. The first model finds an optimal solution for a bistatic sensor network, i.e., exactly one source and one receiver, assuming a cookie cutter sensor model. Then we discuss how INLPs perform on extended networks.

2.3.1 Bistatic Cookie Cutter Sensor Networks

For this model we start from the premise that we already found the set of vertices C, using the Graham scan, for example. Then we formulate the following INLP to find optimal positions for a bistatic sensor network assuming a cookie cutter sensor model.

Indices and Sets:

$t \in T$	targets,
$p \in P = \{s, r\}$	type of sensor, s for source and r for receiver.

Data [units]:

x_t	x coordinate of target t [unitless],
y_t	y coordinate of target t [unitless],
v_t	weight of target t [unitless],
ρ_0	range of the day [miles],
C	vertex set, $C = \{t \in T \mid t \text{ is a vertex}\}$ [unitless],
M_t	penalty for not covering target t [unitless].

Decision Variables:

λ_t^p	coefficient for sensor p's position subject to vertex t,
x_p	x coordinate of sensor p,
y_p	y coordinate of sensor p,
h_t	binary, 1 if target t is not detected, 0 otherwise.

Formulation:

$$\min_{\boldsymbol{\lambda},\boldsymbol{x},\boldsymbol{y},\boldsymbol{h}} \sum_{t\in T} v_t h_t$$

Subject to

$$x_p = \sum_{t\in C} \lambda_t^p x_t \qquad \forall p \in P, \tag{2.5a}$$

$$y_p = \sum_{t\in C} \lambda_t^p y_t \qquad \forall p \in P, \tag{2.5b}$$

$$1 = \sum_{t\in C} \lambda_t^p \qquad \forall p \in P, \tag{2.5c}$$

$$\rho_0 + M_t h_t \geq d_{t,s} \times d_{t,r} \qquad \forall t \in T, \tag{2.5d}$$

$$0 \leq \lambda_t^p \qquad \forall t \in C, p \in P,$$

$$h_t \in \{0,1\} \qquad \forall t \in T.$$

The model forces the positions of the source and receiver to be inside the convex hull by defining them as linear combinations of the vertices in Equations (2.5a) to (2.5c). We introduce the binary decision variable h_t for each target $t \in T$ specifying in Equation (2.5d) whether a target t is detected, $h_t = 0$, or hidden, $h_t = 1$. If t is not covered, then a large number M_t is added to the range of the day, ρ_0, making sure the constraint is still feasible.

Camm, Raturi, and Tsubakitani (1990) show that tight penalty parameters like M_t reduce the time a solver needs to find an optimal solution. In order to make sure the INLP is always feasible, the left-hand side (LHS) of this equation has to be equal to the largest product of distances possible for each target. Since the largest distance from a target position to a point in a convex hull is to one of its vertices, we can set M_t to its square. We can also substract ρ_0 since it is already added on the LHS. Thus we have

$$M_t = \max_{t'\in C} d_{t,t'}^2 - \rho_0. \tag{2.6}$$

It is important to notice that there is generally more than one optimal solution with this model. First of all, interchanging the source and receiver in a bistatic environment does not change the outcome. The Cassini oval drawn by both sensors is still the same. But more importantly, if a sensor is moved slightly in one direction, it may still detect the same targets. And last but not least, there might be a solution detecting different targets, but the same number of targets. Hence, in most scenarios there exist infinitely many solutions.

2.3.2 Other Networks

The model described in Section 2.3.1 only supports bistatic cookie cutter sensor networks. Consequently, there are two ways to extend this model: finding positions for multiple sources and receivers, and expanding to diffuse sensor models. Though models can be formulated, up to this point we are not able to define a model that runs in reasonable time. In a future study, advanced versions of this model can be further investigated.

CHAPTER 3:
Divide Best Sector Algorithm

In this chapter we slightly change the scenario. In an operational setting, it is not unlikely that receivers are already deployed throughout a field by plane or ship. As soon as a hostile underwater asset is suspected in the area of responsibility (AoR), e.g., by passive detection, sources are used to send out pings in order to find the exact position of the intruder (Navy personnel, personal communication, April 2014). As discussed, it is crucial to send out as few sources and subsequent pings as possible.

3.1 Problem Statement

Consider the following scenario: there is a set T of targets with fixed positions that have to be monitored or detected. Furthermore, there is a set R of receivers whose positions we already fixed. The objective is to find optimal positions for a limited number of sources, defined in set S. We do not assume a particular sensor model; rather, we discuss a method that works with all sensor models shown in Figure 1.3.

Figure 3.1 shows a generic scenario with fixed targets and receivers. The z-axis represents the average detection probability assuming a single source is deployed at the respective position. Multiple local maxima in this plot show that the objective function is nonconvex. Nonetheless we now develop an algorithm guaranteed to solve this problem to near global optimality when $|S| = 1$.

3.2 Algorithm Development

In this section we develop the Divide Best Sector (DiBS) algorithm. The algorithm partitions the area of possible solutions into sectors. At each iteration the sector with the highest upper bound for the objective function is further divided into smaller sectors. In Section 2.1 we showed that optimal sensor positions can always be found inside the convex hull spanned by the targets. We slightly relax this condition to the smallest rectangle with edges parallel to the x- and y-axes that contains all targets. This way we can easily divide the area of possible source locations into smaller sectors by slicing horizontally as well as

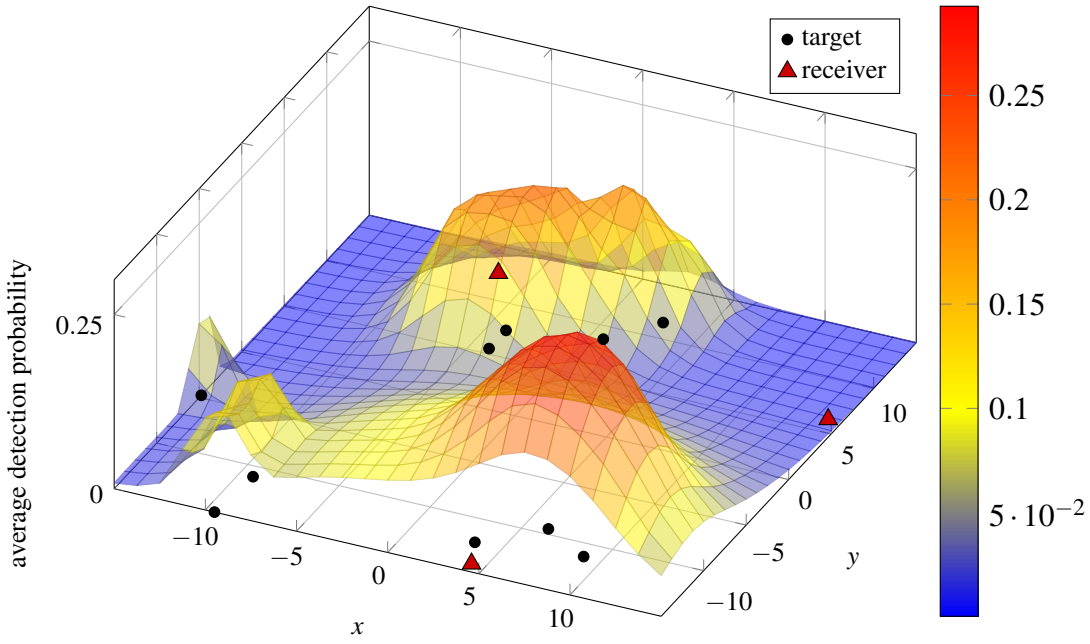

Figure 3.1: **Problem Statement** - The optimal source position with fixed target and receiver locations needs to be found. The z-axis represents the average detection probability assuming a source is deployed at the respective position. Multiple local maxima indicate nonconvexity.

vertically.

Each sector γ of the resulting set of sectors Γ now is evaluated with respect to an upper bound, $\mathrm{uB}(\gamma)$, for the objective function. This is the heart of DiBS and illustrated in Figure 3.2. Since receivers and targets occupy fixed positions in this scenario, the target-receiver distance $d_{t,r}$ is constant. Thus, the equivalent range $\rho_{t,s,r}$ and subsequently the detection probability P_t only depend on the target-source distance $d_{t,s}$. Hence, for every target we determine the hypothetical source position that is closest to the target but still inside γ. These hypothetical source positions are either on the edge of the sector or the target position itself, as seen in Figure 3.2.

With the shortest distance $d_{t,\gamma}$ from target t to its respective hypothetical source position inside sector γ we can compute the highest possible detection probability P_t, assuming the source location is inside γ. Having done this with all targets, we compute the upper bound, $\mathrm{uB}(\gamma)$, according to Table 1.1. The following pseudocode finds the upper bound for a given sector γ, where x_γ^{min} (y_γ^{min}) is the lowest and x_γ^{max} (y_γ^{max}) the highest x (y) coordinate for γ.

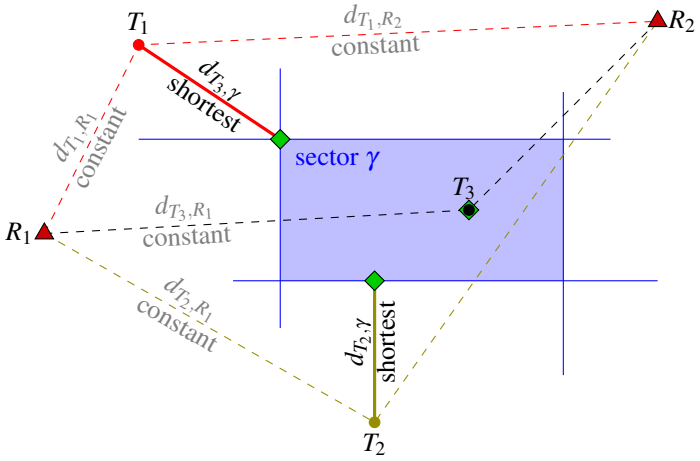

Figure 3.2: **Determine Upper Bound** - For each target t there exists a hypothetical source position inside sector γ that has the shortest distance $d_{t,\gamma}$ to the target. This distance determines the highest possible detection probability with a source position inside γ. Merging the probabilities for all targets results in an upper bound for γ.

procedure UPPER.BOUND(γ)
  for all $t \in T$ **do**
    $x_s \leftarrow \min(\max(x_\gamma^{min}, x_t), x_\gamma^{max})$
    $y_s \leftarrow \min(\max(y_\gamma^{min}, y_t), y_\gamma^{max})$
    $d_{t,\gamma} \leftarrow \sqrt{(x_t - x_s)^2 + (y_t - y_s)^2}$
    compute P_t ▷ use chosen sensor model
  end for
  compute objective value Z ▷ use chosen objective
  return Z
end procedure

In a next step we pick the sector $\gamma \in \Gamma$ in which $\mathrm{uB}(\gamma)$ is maximized and divide it into smaller sectors with respective upper bounds. Consistently repeating this method results in smaller sectors and subsequently tighter upper bounds until a termination condition is met. Various termination conditions are conceivable, such as a maximum sector size, optimality range, etc. The following pseudocode shows the workings of DiBS.

```
1:  procedure DiBS
2:      create initial Γ
3:      optimal ← FALSE
4:      while ¬optimal do
5:          select γ ∈ Γ such that Upper.Bound(γ) is maximal
6:          if termination condition is met then        ▷ use chosen termination condition
7:              optimal ← TRUE
8:          else
9:              Γ′ ← split γ into smaller sectors
10:             Γ ← Γ′ ∪ Γ \ {γ}
11:         end if
12:     end while
13:     return γ
14: end procedure
```

Figure 3.3 shows the example from Figure 3.1 solved with DiBS. It vividly illustrates how sectors with a low upper bound are ignored while promising areas are further explored. For the rest of the chapter we look at various details of DiBS and give recommendations on how to use it most efficiently.

3.3 Termination Conditions

First, we want to make sure, is that the algorithm eventually finishes when it meets a specific condition. This section discusses various termination conditions based on sector size and optimality gap.

3.3.1 Termination By Sector Size

There are two ways to define the sector size: by its area and by its edge lengths. A small area, however, can be the result of a very narrow but long rectangle. This metric is not very useful, because detection probabilities can change greatly along a line, as well as in a long rectangle. Thus, even if a sector with small area has a high upper bound, a sensor placed in this sector may perform badly. Additionally, telling a decision maker to place the source in a $0.01 \times 10,000$ foot rectangle might not be useful in practice.

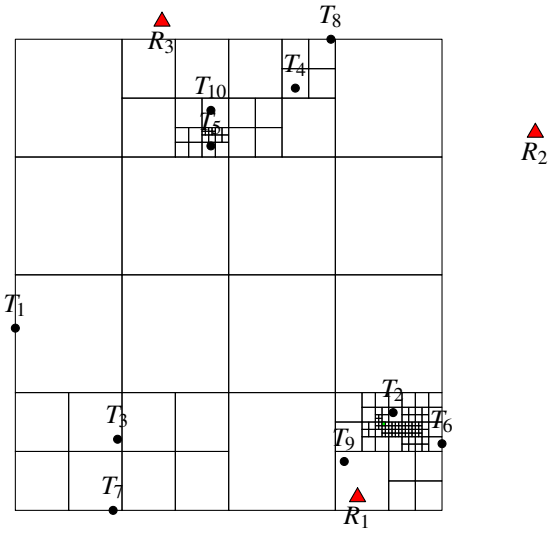

Figure 3.3: **Algorithm Example** - A scenario with ten targets and three receivers is given (see Figure 3.1). DiBS finds the source position with the highest average detection probability by dividing only promising sectors, i.e., sectors with a high upper bound, into smaller ones.

On the other hand, using a maximum edge length strongly constrains possible source locations. Choosing a maximum edge length of ten feet, for example, results in a final sector that has size of at most 10×10 feet, a sufficient precision for helicopter dips. Moreover the change in detection probabilities in such a region is very limited.

Therefore, we disregard using area as the termination condition. For the rest of this chapter, termination by sector size denotes termination when the length of the longest edge of the sector with the highest upper bound is sufficiently small.

3.3.2 Termination By Optimality Gap

Though simple and useful, it turns out that termination by sector size has a drawback. If we request a sector with very short edges, the difference for the upper bounds between adjacent, same-sized sectors becomes negligible. This leads to dividing all sectors with the same size in an area before continuing with smaller ones. This, however, takes the majority of the time and, on top of this, the gain in objective value is almost always negligible.

Hence, we expand the termination condition with a technique used by solvers: we introduce a tolerated optimality gap, ε. By allowing DiBS to stop as soon as it finds a specific sensor

position s' that has an objective value within ε of the upper bound, we are able to stem the described effect. The effectiveness of the optimality gap strongly depends on which position we choose for s'.

We take a closer look at the methods summarized by Table 3.1. A method with fewer potential source positions, like *Center*, does not require high computational power but is subject to missing the optimality gap more often. On the other hand, checking five positions with *Corners+Center* is more precise but increases run time. The last method, *Targets*, is an add-on for the previous ones. The idea is that the detection probability for a single target is replacedmaximizedmaximum only if at least one sensor is at the target's position. Since there are sectors that do not contain any targets, it can only be used in conjunction with another method.

Method	Amount	Remarks
Center	1	uses sector's center point
Corners	4	uses sector's vertices
Corners+Center	5	combines Corners and Center
Targets	0,1,...	uses target positions inside sector

Table 3.1: **Optimality Gap Methods** - These Methods look for a specific source position inside a sector in order to determine the optimality gap. The last method, Targets, has to be used in conjunction with one of the other methods since sectors with no targets inside exists.

The example in Figure 3.4 shows how an optimality gap limits the number of created and evaluated sectors and thus the run time. In this particular problem instance the baseline case created more than 16,000 sectors for a demanded precision of 0.001. Utilizing an optimality gap of $\varepsilon = 0.05$, however, already terminates DiBS after about 200 created sectors with only a minor difference between Center and Corners.

Running more examples, we observe that the gap between Center and Corners always is small. Most of the time Corners terminates with fewer sectors than Center, because it can choose from four potential source positions instead of just one. Center, by contrast, has the better run time. Since Corners+Center combines both methods it always uses the same number of sectors as the method with fewer sectors, but it takes the longest run time. In all tests we ran, we never observed that adding the method Targets had any effect on the outcome except for increasing run time.

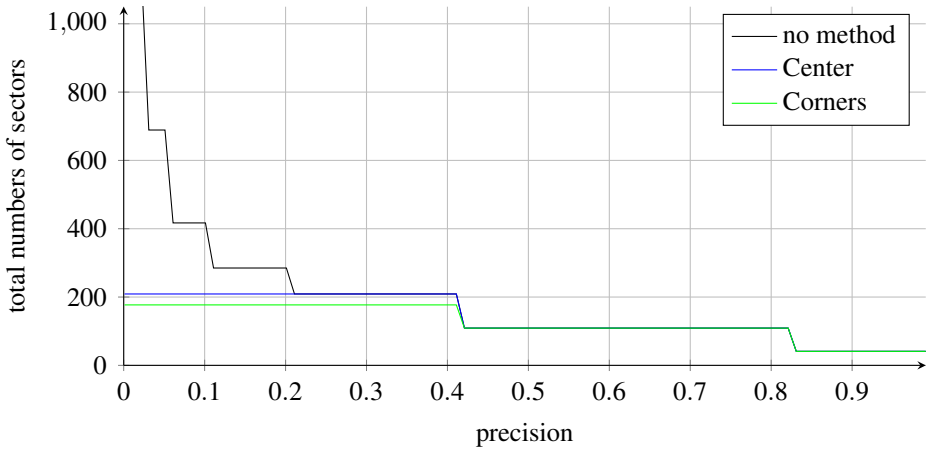

Figure 3.4: **Optimality Gap Example** - While the number of evaluated sectors for the baseline case without optimality gap feature increases with higher demanded precision, using an optimality gap terminates the algorithm earlier.

Including the termination condition with an optimality gap is strongly recommended since it significantly reduces the number of created and evaluated sectors, especially for high precision calculations. Even though the run time per iteration slightly increases, the total run time can reduce to a fraction of the original run time, whereas the choice of method for finding the definite source position s' hardly changes the outcome.

3.3.3 Utilizing Lower Bounds

On closer examination, we realize that the methods described in the previous section provide a lower bound, $\mathrm{lB}(\gamma)$. This section discusses whether we can use lower bounds to disregard sectors. Additionally, we show the performance of DiBS using lower bounds.

Consider discarding a hypothetical sector γ' whose upper bound, $\mathrm{uB}(\gamma')$, is less or equal than the lower bound, $\mathrm{lB}(\gamma)$, of another sector γ. Surprisingly, this is not necessary. Since $\mathrm{lB}(\gamma)$ is the objective value for a definite source position in γ, it is always included in one sector of γ's split. Hence there is always a sector out of γ's split that still has a higher upper and lower bound than γ'. Thus, γ' will never be chosen as the best sector, and therefore, discarding does not reduce the total number of sectors created.

It does, however, reduce the number of sectors in a list that has to be ordered by upper bounds. Most programming languages use fast sorting algorithms like *Quicksort*. So or-

dering a large list compared to other operations in DiBS is negligible.

Nevertheless, we can use the lower bound to indirectly measure the goodness of the upper bound. A tight upper bound is important for faster identification of a solution. Figure 3.5 illustrates the change of gap between upper and lower bound throughout the execution of DiBS. Here, the right plot shows the underlying trend of a decreasing gap the longer the algorithm runs. Jumps to lower values for the lower bound occur when the next sector with highest upper bound is much larger than the previous sector and therefore can provide a bigger gap between lower and upper bound. The left plot verifies our suspicion that smaller sectors yield smaller gaps and therefore tighter upper bounds.

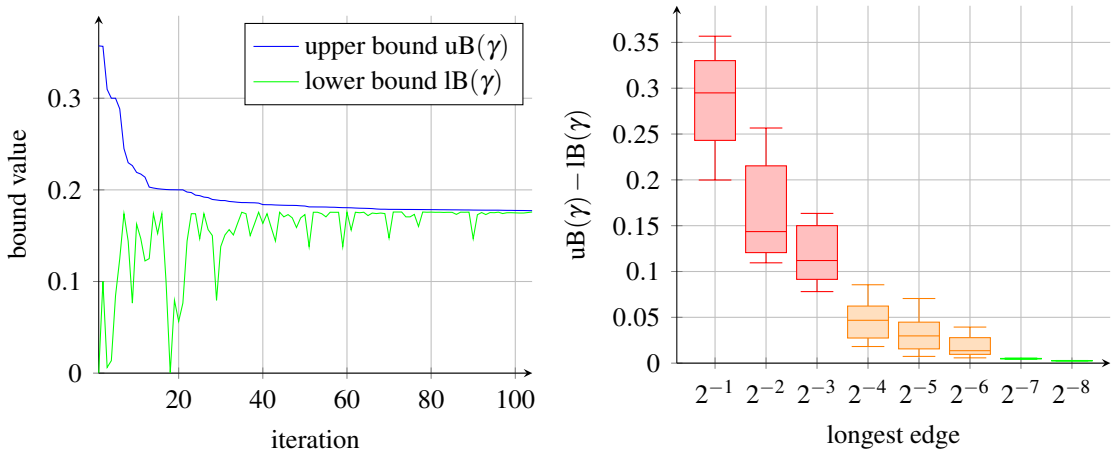

Figure 3.5: **DiBS Goodness** - With increasing number of iterations the gap between upper and lower bound decreases (left plot). Steep cuts on lower bound values occur on iterations where a small sector is sufficiently split and DiBS jumps back to a larger sector and thus larger gap. The right plot mirrors the relationship between longest edge size as a multiple of longest initial edge and gap size. Smaller sectors yield tighter upper bounds.

Summarizing, we conclude that lower bounds do not accelerate DiBS. They can, however, be used to show that upper bounds are getting tighter along the algorithm. Hence, the algorithm quickly achieves a small optimality gap, which can be used to terminate DiBS early as shown in Section 3.3.2.

3.4 Sector Splitting

Since the algorithm begins with creating the initial set of sectors Γ, this procedure and subsequently the dividing into smaller sectors is the next feature of DiBS into which we

take a closer look. The question is how many vertical and horizontal slices are efficient with respect to complexity and run time? First we investigate a fixed number of slices and expand later on to dynamic splitting methods.

3.4.1 Fixed Splitting

More slices reduce sector sizes faster and therefore meet the termination condition in fewer iterations of DiBS. Each iteration, however, requires more computations and thus more time. The plots in Figure 3.6 show the algorithm's performance for a particular problem instance using various settings for initial division and further splitting. The derived source position is always the same for all settings. As assumed, the number of iterations decreases quickly with more initial sectors and even more with a higher number of new sectors per iteration. The positive effect, however, stagnates very quickly.

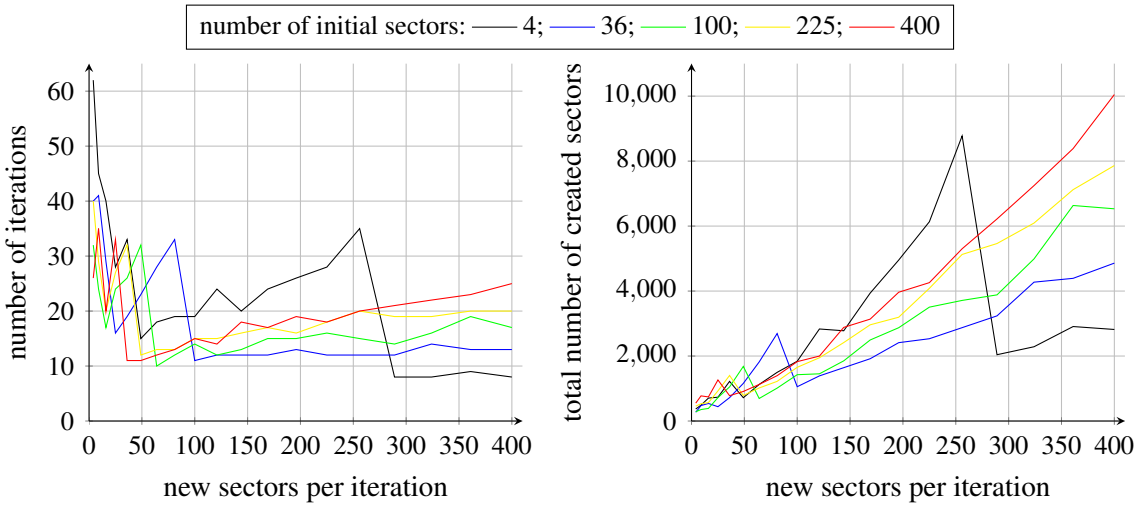

Figure 3.6: **Sector Splitting** - Algorithm's performance with different splitting settings. Splitting into many sectors per iteration increases the total number of sectors and subsequently the run time of the algorithm. The size of the initial set of sectors also affects run time especially for high numbers of new sectors per iteration. The reduction in number of total iterations is not as decisive.

On the other hand, the right plot in Figure 3.6 reflects the increasing number of created sectors. While the change in the number of iterations is between 8 and 62, the values for the number of created sectors range from 270 to over 10,000. Running DiBS on a test machine shows that the run time is directly related to the number of created sectors, e.g., 0.9 seconds for an initial and further split of 4 sectors versus 28.7 seconds for an initial and

further split of 400 sectors. The number of iterations does not seem to have any effect on the run time.

The results are decisive for newly created sectors: dividing the best sector into fewer new sectors greatly decreases run time and improves algorithm efficiency. A reason for this observation lies in dividing small sectors. We know from Section 3.3.3 that small sectors have tighter upper bounds. Hence, the variance in upper bounds of their split sectors is small. So after splitting a small sector into a set of many small sectors Γ', it is very likely that the algorithm has to evaluate and, even worse, split most of the sectors in Γ' before it is able to continue with a smaller sized sector. This behavior creates many sectors that are unnecessarily evaluated and therefore waste resources. A different approach is discussed in Section 3.4.2.

The creation of the initial set of sectors also has an effect on the total number of created sectors. Especially at high numbers of new sectors per iteration it becomes a critical factor, as seen in Figure 3.6. It appears a smaller initial set performs more efficiently.

In most cases, starting with a small set of sectors and slicing the best sector into a small number of new sectors is most efficient and performs best. We recommend to always start with this setting and expand as circumstances require.

3.4.2 Dynamic Splitting

We discovered in Section 3.4.1 that splitting into many sectors per iteration produces a lot of overhead, especially at small sector sizes. The question arises whether one can reduce the number of created sectors via dynamic splitting. In dynamic splitting, large sectors or sectors with a high optimality gap are split into more sectors than small sectors or sectors with tighter upper bounds. We leave that question open for future work.

3.4.3 Plane Rotation

The DiBS algorithm starts with finding the smallest rectangle with edges parallel to the x- and y-axes that contains all targets. We know from Section 2.1 that the optimal sensor position lies inside the convex hull spanned by the targets. Hence, it is possible that the resulting rectangle contains a large overhang which is subject to being evaluated unnecessarily. This section discusses the use of the minimum-area rectangle without constrained

edges and how to rotate the plane, such that DiBS still can be applied.

Freeman and Shapira (1975) prove that the minumum-area rectangle encasing a convex hull has a side collinear with one of the edges of the convex hull, like the examples in Figure 3.7. Hence we choose the rectangle with smallest area out of the $|C|$, i.e., the number of vertices of the convex hull, rectangles we can form. Using the Graham scan to find C has the advantage that the vertices are already ordered counter clockwise. Therefore, we can use this output to easily construct the edges of the convex hull.

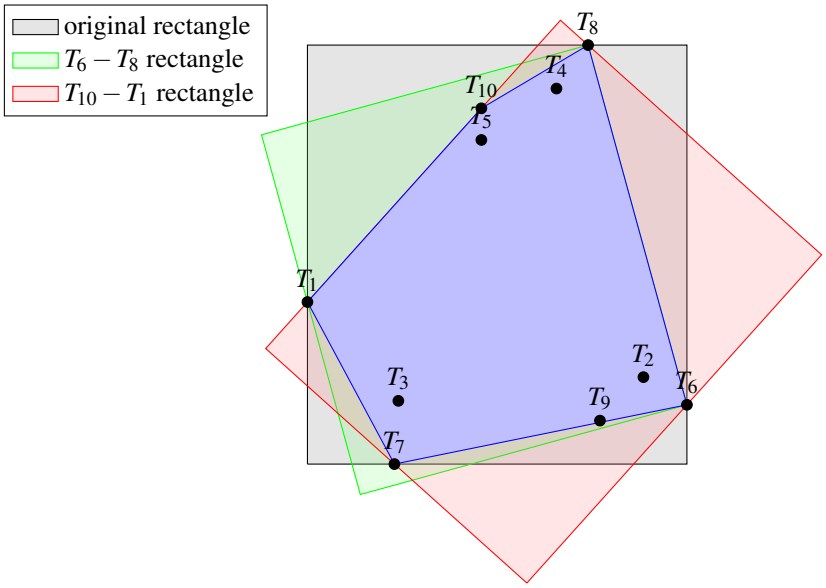

Figure 3.7: **Minimum-Area Rectangle Example** - The plot shows the original rectangle as well as two rectangles that have one side collinear with one side of the convex hull. The $T_6 - T_8$ rectangle is the minimum-area rectangle that encases all targets.

The example in Figure 3.7 shows the original rectangle produced by DiBS as well as the rectangles with the smallest and the largest area that have one side collinear with one of the edges of the convex hull. The respective areas in square units are

$$\text{original: } 636, \qquad T_6 - T_8\text{: } 505.375, \qquad T_{10} - T_1\text{: } 619.420.$$

This example shows that using the minimum-area rectangle can greatly reduce the overhang. Having determined the minimum-area rectangle, we need to rotate the plane such that the edges of the rectangle are parallel to x- and y-axis. An efficient way to do this is

applying a linear transformation to the sets of target and receiver locations (Leon, 2010, p. 166ff.). Let $t_0 - t_1$ be the edge that is collinear with the minimum-area rectangle. Then the linear transformation is represented by the matrix

$$A = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \text{ where } \theta = \tan^{-1}\left(\frac{y_{t_1} - y_{t_0}}{x_{t_1} - x_{t_0}}\right).$$

The matrix multiplication AX provides the new coordinates for targets and receivers, where the first and second row of X holds the x and y coordinates respectively.

Since plane rotation can have a huge impact on the size of the overhang, it should always be considered before running DiBS. There is also the fact that Graham scan and linear transformation are very efficient and therefore executed quickly.

3.4.4 Handling Uneven Rectangles

The next improvement on sector splitting aims for rectangles with uneven edge lengths. Obviously the short edge reaches its termination condition faster than the long edge if DiBS performs the same number of slices in both directions at each iteration. Since termination by sector size considers only the longest edge, the algorithm continues dividing sectors in both directions. Hence it creates more sectors than needed.

There are two ways to address this problem: stopping splits in directions that meet the termination condition and finding an initial split that has only square sectors. The first method is easy to implement by setting the number of slices for edges that meet the termination condition to zero. There is, however, already a problem with uneven rectangles before the short edge meets the termination condition. As opposed to squares with the same area, especially long rectangles are more likely to have a big variance in detection probabilities (see Section 3.3.1). This subsequently leads to looser upper bounds.

The second approach bypasses this issue by starting with squares at the first iteration. We already discovered in Section 3.4.1, that a small set of initial sectors performs more efficiently. So we are looking for a method that creates a low number of squares that covers an uneven rectangle.

Figure 3.8 shows two methods to create initial squares. For the rest of this section, we

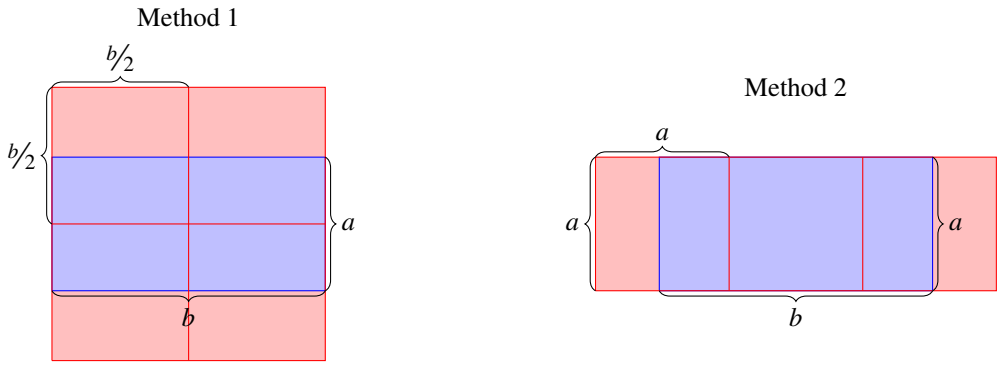

Figure 3.8: **Creating initial squares** - The first method uses half of the long edge b for square edge length, while method 2 uses the short edge. The red shaded area reflects the created overhang when using only squares for the initial split.

assume that a and b are the short and the long edge respectively. The first method cuts the long side b in half and takes the result as the edge length for the squares. The second method, on the other hand, uses the short edge a as the edge length for the squares. Both methods, however, create an overhang, shaded red in the figure, that hold non-optimal solutions. The overhangs are calculated as follows:

$$\text{For method 1: } \Omega_1 = b^2 - ab. \tag{3.1}$$

$$\text{For method 2: } \Omega_2 = \begin{cases} 0 & \text{if } a \mid b, \\ [a - (b \bmod a)]a & \text{otherwise.} \end{cases} \tag{3.2}$$

In order to keep the unneeded area minimal, we always choose the method with the smallest overhang. Table 3.2 lists two examples with different choices of the method used.

a	b	$r = a/b$	Method 1 Overhang	Method 2 Overhang	Select
4	5	0.8	$5^2 - 4 \times 5 = 5$	$[4 - (5 \bmod 4)]4 = 12$	Method 1
2	5	0.4	$5^2 - 2 \times 5 = 15$	$[2 - (5 \bmod 2)]2 = 2$	Method 2

Table 3.2: **Examples for Overhang** - For the first example, method 1 has the smaller overhang, hence we select it. For the second example, however, we choose method 2 because of its smaller overhang.

Going a step farther, we want to see whether we are able to decide which method to use by solely looking at the ratio between long and short edge.

Theorem 3.1. *Method 1 has less overhang than method 2 if and only if* $\frac{1}{\sqrt{2}} \leq r < 1$, *where* $r = a/b$.

Proof. Without loss of generality, we assume $b = 1$. Then $0 \leq r = a \leq 1$. Furthermore the overhangs for both methods compute as follows:

$$\Omega_1 = b^2 - ab = 1 - r,$$
$$\Omega_2 = \begin{cases} 0 & \text{if } a \mid b, \\ [a - (b \bmod a)]a = [r - (1 \bmod r)]r & \text{otherwise.} \end{cases}$$

Method 1 has less overhang if and only if $\Omega_1 - \Omega_2 < 0$, which is visualized in Figure 3.9. With a ratio of $r = 1$, i.e., both sides have the same length, both methods do not have an overhang since a and b already form a square. Discontinuities occur on positions where a divides b, such as ½, ⅓, ¼ and so on. Here, method 2 produces no overhang. Moreover, by inspection, we see $\Omega_1 - \Omega_2 > 0$ for $0 \leq r \leq 0.5$.

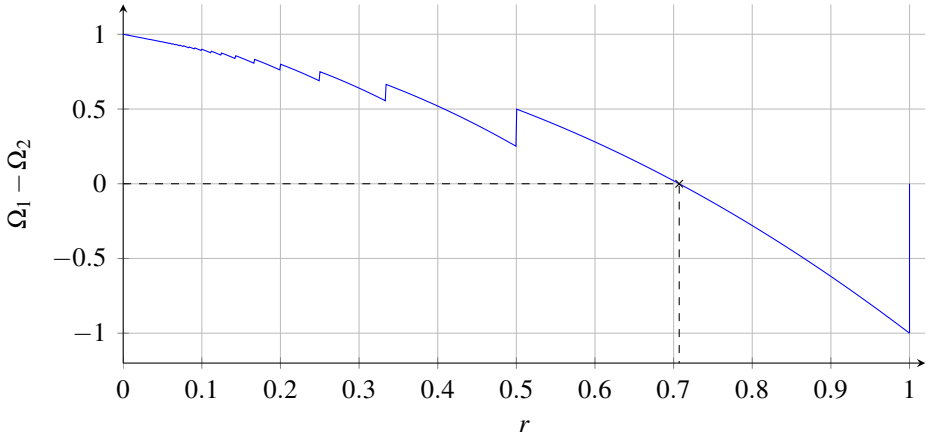

Figure 3.9: **Theorem 3.1 Proof** - The figure shows the difference between method 1's overhang, Ω_1, and method 2's overhang, Ω_2, with respect to the ratio $r = a/b$. A positive value indicates a smaller overhang for method 2, while method 1 is preferred at negative values.

Hence, we are looking for a root in $0.5 < r < 1$, where $a \nmid b$ and $1 \bmod r = 1 - r$. Thus

$$\begin{aligned}\Omega_1 - \Omega_2 &= 1 - r - [r - (1 \bmod r)]r \\ &= 1 - r - [r - (1 - r)]r \\ &= 1 - 2r^2,\end{aligned}$$

of which the only nonnegative root is $\frac{1}{\sqrt{2}}$. □

When DiBS splits a sector, it does not recognize whether or not a newly created sector is in the overhang. An additional check removes those sectors and prevents selecting them as best sector for further splitting.

3.5 Finding Multiple Sensor Locations

A way to increase the number of possible applications of DiBS is increasing the number of sensor positions optimized. This section investigates an iterative method as well as a simultaneous approach.

3.5.1 Placing Sources Iteratively

We consider a set S of sensors, that has to be optimally deployed in a scenario with fixed targets and receivers. Furthermore, we assume a cookie cutter sensor model. The position for the first source is determined by run DiBS without modifications. In a next step we remove all targets that are detected by the algorithm's solution and running DiBS again for the second sensor position. We repeat those step until either all targets are detected or all sources are set. The following pseudocode illustrates the iterative method.

```
1: procedure ITERATIVE.METHOD
2:     while |T| > 0 ∧ |S| > 0 do
3:         select s ∈ S
4:         run DiBS
5:         T ← T \ {detected targets}
6:         S ← S \ {s}
7:     end while
8: end procedure
```

Since we place sources in a greedy approach, the resulting solution generally is not optimal. Notwithstanding, this algorithm reveals some information about the optimal solution. First of all, in case all targets are detected by this method, the solution indeed is optimal.

Otherwise, we can use the outcome to define a lower and upper bound for this problem by applying an observation made by Craparo, How, and Modiano (2011). Leveraging submodularity in a similar coverage setting, the authors determine that an iterative algorithm has an optimality gap of at most $1 - \mathrm{e}^{-1} \approx 0.632$. Hence, we define the lower bound, lB, as the outcome of the iterative approach and the upper bound

$$\begin{aligned} \mathrm{uB} &= \left\lfloor \frac{\mathrm{lB}}{1-\mathrm{e}^{-1}} \right\rfloor \\ &\approx \lfloor 1.582 \times \mathrm{lB} \rfloor . \end{aligned}$$

We conclude that even though the iterative method does not guarantee an optimal solution, it is a quick way to get an estimate about the quality of other solutions. Combined with our observations from Section 2.2, we are able to narrow down the optimal solution even more.

3.5.2 Placing Sources Simultaneously

In contrast to the previous method, a different approach is placing multiple sources simultaneously. In doing so, we place each source in an arbitrary sector and evaluate upper bounds of occupied sectors. Merging those upper bounds yields the value for the objective function. In order to exhaust all possible solutions, we evaluate all combinations of sectors. We have to take into account that multiple sources might occupy the same sector and that the order of sources is indifferent. Hence, by definition the number of combinations is

calculated by combinations of multisets (Brualdi, 2010, p. 52ff.) as

$$\binom{|S|+|\Gamma|-1}{|S|} = \frac{(|S|+|\Gamma|-1)!}{|S|!(|\Gamma|-1)!} \tag{3.3}$$

where $|S|$ is the number of sensors and $|\Gamma|$ the number of sectors in a particular iteration. The base case with one source represents the original DiBS where the number of combinations equals the number of sectors, i.e., placing the source once in each sector. With two sources and four sectors we already have the following ten combinations.

$$\begin{array}{lllll} 1:\{\gamma_1,\gamma_1\} & 2:\{\gamma_1,\gamma_2\} & 3:\{\gamma_1,\gamma_3\} & 4:\{\gamma_1,\gamma_4\} & 5:\{\gamma_2,\gamma_2\} \\ 6:\{\gamma_2,\gamma_3\} & 7:\{\gamma_2,\gamma_4\} & 8:\{\gamma_3,\gamma_3\} & 9:\{\gamma_3,\gamma_4\} & 10:\{\gamma_4,\gamma_4\} \end{array}$$

The number of combinations grows quickly with the number of sectors and sources which is visualized in Figure 3.10. While placing just a few sources simultaneously might still be manageable, using this approach with many sensors quickly becomes an issue. This is aggravated by the fact that selecting and dividing multiple sectors per iteration increases the number of sectors and subsequently the number of combinations even more quickly. A future study might find ways to reduce that number or find another method to deal with multiple sources.

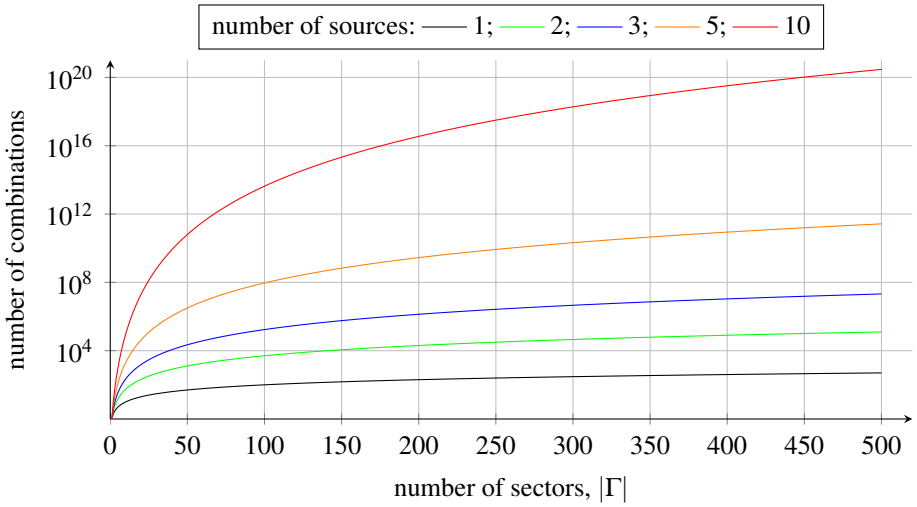

Figure 3.10: **Sector Combinations** - This plot displays the number of combinations that have to be evaluated with respect to the number of sectors and sources. The base case with one source represents the original DiBS.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Conclusion

We started by developing some insights that help bound the problem. We showed that placing sensors inside the convex hull that encases all targets results in detection probabilities as least as good as outside the hull. This important observation limits the area where we have to search for optimal sensor positions regardless of the sensor model we use.

Assuming a cookie cutter sensor model, we can utilize clusters of RDCs. The number of elements in the minimal set of clusters, $\widetilde{G}$, represents a lower bound on the number of sensors required to cover all targets. In addition, with a fixed number of sensors, n, we are able to determine an upper bound on the number of targets by selecting n clusters from $\widetilde{G}$, such that the number of contained targets is maximized. Other circles, such as RCs and SCs, are means to narrow down possible sensor positions assuming the location of one kind of sensor is fixed.

In a subsequent step, we formulated an INLP to find optimal positions for a bistatic cookie cutter sensor network. Up to this point, expanding this model to multiple sensors or other sensor models does not provide a solution in reasonable time. Hence, we developed the DiBS algorithm.

DiBS assumes that target and receiver positions are fixed and determines the optimal position for a single source regardless of the applied sensor model. By dividing sectors with the highest upper bound for the objective function, it eventually reaches a termination condition like maximum size of a sector's longest edge or a maximum optimality gap.

Further investigations of the DiBS algorithm's details resulted in the following observations and recommendations. The number of new sectors created at each iteration as well as the number of initial sectors should be kept small in order to reduce the total number of created and evaluated sectors and subsequently the run time of DiBS.

Rotating the plane such that the edges of the minimum-area rectangle encasing all targets are parallel to the x- and y-axis results in a smaller needless overhang. Because of the high

efficiency to find the minimum-area rectangle and to rotate the plane, this improvement should always be considered.

Furthermore, we discussed two ways to handle uneven rectangles that can lead to creating and evaluating many unnecessary sectors. Both methods approach this issue differently. Their efficiency depends on the given scenario. We recommend, however, to apply one of the methods to reduce the algorithm's run time.

Finally, we illustrate two methods to apply DiBS to problems where we have to find multiple source locations. The faster iterative method assuming a cookie cutter sensor model provides lower and upper bounds on the number of detected targets. The simultaneous approach, however, quickly creates a large number of combinations that have to evaluated. A future study can develop ways to reduce the number of combinations or find other means to find multiple source locations using DiBS.

APPENDIX A:
Excel Analysis And Decision Tool

For a better understanding and visualization of the observations from Chapter 2 , we developed an analysis and decision tool in Microsoft Excel. It provides a number of features for a bistatic cookie cutter sensor model, e.g., showing the Cassini oval.

The *Data* worksheet shown in Figure A.1 contains the scenario data such as target positions and range of the day. Furthermore, it shows the positions of source and receiver and determines the detection probability for each target based on target-source and target-receiver distance.

Range of the Day	
ρ_0	3.5

Source	x	y
Source1	1.6	3.7

Receiver	x	y
Receiver 1	-4.4	6.3

Update Target Information

Load Targets from .CSV

Object	x	y	Src Dist	Rcv Dist	detected
T1	-15	-4.75	18.6269294	15.3121684	FALSE
T2	6.25	-9.5	13.9950884	19.054199	FALSE
T3	-9.25	-11	18.2705364	17.9669836	FALSE
T4	0.75	8.75	5.12103505	5.70306935	FALSE
T5	-4	5.5	5.88217647	0.89442719	TRUE
T6	9	-11.25	16.681202	22.0808175	FALSE
T7	-9.5	-15	21.746264	21.9020547	FALSE
T8	2.75	11.5	7.88431988	8.84095583	FALSE
T9	3.5	-12.25	16.0627675	20.1621551	FALSE
T10	-4	7.5	6.76756973	1.26491106	TRUE

Figure A.1: **MSN Tool Data Sheet** - The *Data* worksheet contains information about targets, sensors and range of the day. It provides methods to change the scenario.

There are two ways to change the target data: by changing the values in the table or by loading a comma-separated values (CSV) file. Using the first method, we have to press the *Update Target Information* button after all changes were made in order to apply the changes. The second method initiated by clicking *Load Targets from .CSV* does not require this step. The following example shows the required structure for a CSV file.

```
T1,-15,-4.75
T2,6.25,-9.5
⋮
```

Figure A.2 shows the *Chart* worksheet that visualizes the scenario. Here, we are able to show or hide various features. The scrollbars on the top and on the right move either source or receiver depending on what is chosen on the left. Detected targets are represented by empty circles; otherwise, targets are represented by filled circles. The *Data* box contains some general information while the *Selection* box provides a short report about each element in the chart based on the mouse position.

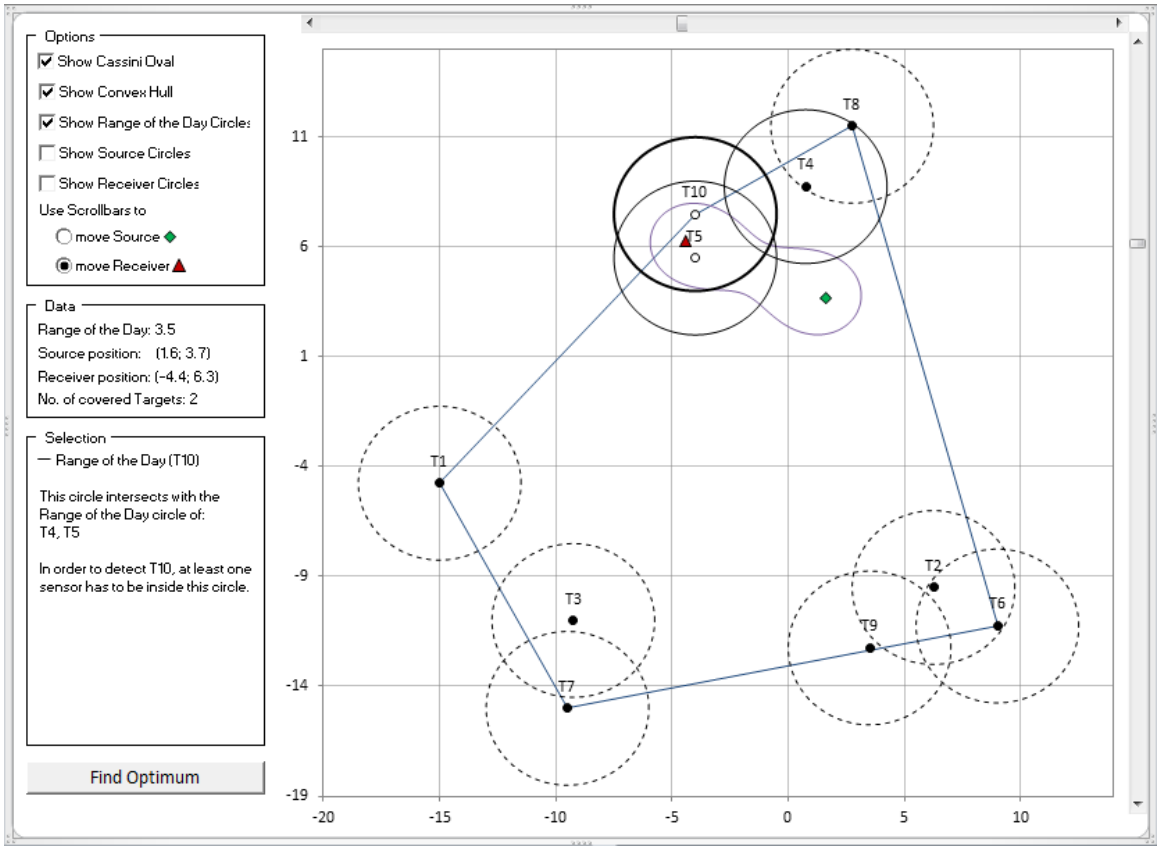

Figure A.2: **MSN Tool Chart** - The scrollbars move either source or receiver based on the selection on the right. The *Selection* box provides additional information about shown objects in the chart when hovering with the mouse. In this example target T_{10}'s RDC is selected.

The *Find Optimum* button is without functionality in this version of the tool. Here, the INLP from Section 2.3 can be integrated in future work.

APPENDIX B:
R Implementation Of DiBS

```r
DiBS <- function(targets, receivers, rho_0, longest_edge, verbose = TRUE, obj = 'avg',
                 sensor = 'fermi', b = 0.25, gap_method = 'none', gap = 0.05) {
  # Main function to run DiBS
  #
  # Args:
  #   targets: Data frame with object, x, y and value.
  #   receivers: Data frame with object, x and y.
  #   rho_0: Range of the day.
  #   longest_edge: Termination condition.
  #   verbose: If TRUE, prints progress. Default is TRUE.
  #   obj: Objective. Possible values: 'avg', 'min'. Default is 'avg'.
  #   sensor: Sensor model. Possible values: 'cookie', 'fermi', 'exp'. Default is 'fermi'.
  #   b: Diffusivity parameter for Fermi model. Default is 0.25.
  #   gap_method: Optimality gap method. Possible values: 'none', 'center', 'corners', '
      corners.center'. Default is 'none'.
  #   gap: Accepted optimality gap. Default is 0.05.
  #
  # Returns:
  #   Final set of sectors, where the first row yields the solution.
  terminate <- FALSE; iteration <- 0
  data <- .create.data(targets, receivers)
  sectors <- .create.sectors(data, rho_0, obj, sensor, b, 'initial')
  while(!terminate) {
    iteration <- iteration + 1
    sectors <- sectors[order(-sectors$upper.bound, sectors$longest.edge),]
    gamma <- sectors[1,]
    gamma$lower.bound <- .lower.bound(data, gamma, rho_0, obj, sensor, b, gap_method)
    gamma$optimality <- gamma$lower.bound / gamma$upper.bound
    if (gamma$longest.edge <= longest_edge | 1 - gamma$optimality <= gap)
      terminate <- TRUE
    else {
      if (verbose) {cat(paste0('Iteration ', iteration, ':\n')); print(gamma); cat('\n')}
      if (gamma$xmax - gamma$xmin <= longest_edge) xs <- c(gamma$xmin, gamma$xmax)
      else xs <- seq(gamma$xmin, gamma$xmax, length.out = 3)
      if (gamma$ymax - gamma$ymin <= longest_edge) xs <- c(gamma$xmin, gamma$xmax)
      else ys <- seq(gamma$ymin, gamma$ymax, length.out = 3)
      sectors.prime <- .create.sectors(data, rho_0, obj, sensor, b, xs = xs, ys = ys,
                                       name = paste0('iter', iteration))
      sectors <- rbind(sectors[-1,], sectors.prime)
    }
  }
  cat(paste0('Finished after ', iteration, ' iterations (evaluated ',
             nrow(sectors) + iteration - 1, ' sectors):\n'))
```

```
  print(gamma)
  return (sectors)
}

.create.data <- function(targets, receivers) {
  # Creates the data frame used by DiBS containing target locations,
  # values and distances to receivers
  #
  # Args:
  #   targets: Data frame with object, x, y and value.
  #   receivers: Data frame with object, x and y.
  #
  # Returns:
  #   Created data frame.
  data <- targets[,c('x','y','value')]
  row.names(data) <- targets$object
  data[paste0('dist.', as.factor(receivers$object))] <-
    sapply(row.names(receivers), function(r)
      sapply(row.names(targets), function(t)
        sqrt((targets[t,'x']-receivers[r,'x'])^2+(targets[t,'y']-receivers[r,'y'])^2)))
  return (data)
}

.create.sectors <- function(data, rho_0, obj, sensor, b, name, xs = NULL, ys = NULL) {
  # Creates sectors from the given data and evaluates
  # each sector's longest edge and upper bound. Uses xs and ys as
  # slices. If xs and ys are NULL then it creates the initial split.
  #
  # Args:
  #   data: Data set created by function .create.data().
  #   rho_0: Range of the day
  #   obj: Objective.
  #   sensor: Sensor Model.
  #   b: Diffusivity parameter for Fermi model.
  #   name: identifier for the set of sectors
  #   xs: Vector of vertical slices. Default is NULL.
  #   ys: Vector of horizontal slices. Default is NULL.
  #
  # Returns:
  #   Data frame with sector coordinates and upper bounds.
  if (is.null(xs)) xs <- seq(min(data$x), max(data$x), length.out=3)
  if (is.null(ys)) ys <- seq(min(data$y), max(data$y), length.out=3)
  sectors <- data.frame(
    sector = paste0(name, '_', 1:((length(xs) - 1) * (length(ys) - 1))),
    xmin   = rep(xs[-length(xs)], length(ys) - 1),
    xmax   = rep(xs[-1],          length(ys) - 1),
    ymin   = rep(ys[-length(ys)], each = length(xs) - 1),
    ymax   = rep(ys[-1],          each = length(xs) - 1))
  sectors$longest.edge <- pmax(sectors$xmax - sectors$xmin, sectors$ymax - sectors$ymin)
```

```
  sectors$upper.bound <- sapply(1:nrow(sectors), function(s) {
    switch(obj, avg = value <- 0, min = value <- Inf,
           stop(paste0('objective "', obj, '" not known')))
    if (distr == 'cookie') value <- 0
    for(t in 1:nrow(data)) {
      x <- min(max(sectors$xmin[s], data$x[t]), sectors$xmax[s])
      y <- min(max(sectors$ymin[s], data$y[t]), sectors$ymax[s])
      Pt <- .Pt(data, t, x, y, rho_0, sensor, b)
      if(sensor == 'cookie') {
        value <- value + Pt*data[t, 'value']
      } else {
        switch(obj,
               avg = value <- value + data[t, 'value'] * Pt / nrow(data),
               min = value <- min(value, data[t, 'value'] * Pt)
        )
      }
    }
    return (value)
  })
  return (sectors)
}

.Pt <- function(data, t, x, y, rho_0, sensor, b) {
  # Calculates detection probability Pt for target t with source at position (x, y)
  #
  # Args:
  #   data: Data set created by function .create.data().
  #   t: target
  #   x: source's x coordinate
  #   y: source's y coordinate
  #   rho_0: Range of the day
  #   sensor: Sensor Model.
  #   b: Diffusivity parameter for Fermi model.
  #
  # Returns:
  #   Detection Probability Pt
  d.ts <- sqrt((x - data$x[t])^2 + (y - data$y[t])^2)
  Pt <- 1
  for(r in 4:ncol(data)) {
    rho.tsr <- sqrt(data[t,r] * d.ts)
    switch(sensor,
           cookie = if (rho.tsr <= rho.0) {Pt <- 0; break},
           fermi  = Pt <- Pt * (1 - 1 / (1 + 10^(((rho.tsr / rho.0) - 1) / b))),
           exp    = Pt <- Pt * (1 - 10^(-0.30103 * rho.tsr / rho.0)),
           stop(paste0('distribution "', distr, '" not known'))
    )
  }
  return (1-Pt)
}
```

```
141
142 .lower.bound <- function(data, gamma, rho_0, obj, sensor, b, gap_method) {
143   # calculates a lower bound for sector gamma
144   #
145   # Args:
146   #   data: Data set created by function .create.data().
147   #   gamma: Sector data created in function DiBS()
148   #   rho_0: Range of the day.
149   #   longest_edge: Termination condition.
150   #   obj: Objective. Possible values: 'avg', 'min'.
151   #   sensor: Sensor model. Possible values: 'cookie', 'fermi', 'exp'.
152   #   b: Diffusivity parameter for Fermi model.
153   #   gap_method: Optimality gap method. Possible values: 'none', 'center', 'corners', '
      corners.center'.
154   #
155   # Returns:
156   #   Lower bound for sector gamma.
157   switch(gap_method,
158          none           = return (0),
159          center         = {
160            potentials <- data.frame(x=mean(c(gamma$xmin, gamma$xmax)),
161                                     y=mean(c(gamma$ymin, gamma$ymax)))
162          },
163          corners        = {
164            potentials <- data.frame(x=c(gamma$xmin, gamma$xmax, gamma$xmin, gamma$xmax),
165                                     y=c(gamma$ymin, gamma$ymin, gamma$ymax, gamma$ymax))
166          },
167          center.corners = {
168            potentials <- data.frame(x=c(gamma$xmin, gamma$xmax, gamma$xmin, gamma$xmax,
169                                         mean(c(gamma$xmin, gamma$xmax))),
170                                     y=c(gamma$ymin, gamma$ymin, gamma$ymax, gamma$ymax,
171                                         mean(c(gamma$ymin, gamma$ymax))))
172          },
173          stop(paste0('gap method "', gap_method, '" not known'))
174   )
175   potentials$value <- sapply(1:nrow(potentials), function(p) {
176     switch(obj, avg = value <- 0, min = value <- Inf,
177            stop(paste0('objective "', obj, '" not known')))
178     if (distr == 'cookie') value <- 0
179     for(t in 1:nrow(data)) {
180       Pt <- .Pt(data, t, potentials$x[p], potentials$y[p], rho_0, sensor, b)
181       if(sensor == 'cookie') {
182         value <- value + Pt*data[t, 'value']
183       } else {
184         switch(obj,
185                avg = value <- value + data[t, 'value'] * Pt / nrow(data),
186                min = value <- min(value, data[t, 'value'] * Pt)
187         )
188       }
```

```
    }
    return (value)
  })
  potentials <- potentials[order(-potentials$value),]
  return (potentials$value[1])
}

# Example
# The next examples demonstrates the use of DiBS.
targets <- data.frame(
  object = c(  'T1',  'T2',  'T3',  'T4',  'T5',  'T6',  'T7',  'T8',  'T9', 'T10'),
  x      = c(-15.00,  6.25,- 9.25,  0.75,- 4.00,  9.00,- 9.50,  2.75,  3.50,- 4.00),
  y      = c(- 4.75,- 9.50,-11.00,  8.75,  5.50,-11.25,-15.00, 11.50,-12.25,  7.50),
  value  = 1)
receivers <- data.frame(
  object = c(  'R1',  'R2',  'R3'),
  x      = c(  4.25, 14.25,- 6.75),
  y      = c(-14.25,  6.25, 12.50))

# 1. fermi model, maximize average detection probability, longest edge <= 0.25
sectors <- DiBS(targets, receivers, rho_0 = 3, longest_edge = 0.25)

# 2. cookie cutter model, longest edge <= 0.1
sectors <- DiBS(targets, receivers, rho_0 = 3, longest_edge = 0.1, sensor='cookie')

# 3. like 1. but longest edge <= 0.001, use center for lower bound
sectors <- sectors <- DiBS(targets, receivers, rho_0 = 3, longest_edge = 0.001,
                           gap_method = 'center', gap = 0.01)
```

DiBS Implementation - This code shows a basic DiBS implementation in R. It features all sensor models mentioned in Section 1.2 as well as longest edge and optimality gap termination. It assumes that plane rotation has been applied prior to running DiBS. Uneven rectangles are handled by stopping sector splitting in the direction that reaches the longest edge termination condition.

THIS PAGE INTENTIONALLY LEFT BLANK

Supplementals

- Computer Code of the analysis and decision tool – to be used with Appendix A.
- Computer Code of the DiBS algorithm implemented in R – to be used with Appendix B.

The supplementals are available at Dudley Knox Library of the Naval Postgraduate School in Monterey, CA.

THIS PAGE INTENTIONALLY LEFT BLANK

References

Brualdi, R. A. (2010). *Introductory combinatorics* (5th ed.). Upper Saddle River, NJ: Prentice Hall.

Camm, J. D., Raturi, A. S., & Tsubakitani, S. (1990). Cutting big M down to size. *Interfaces*, *20*(5), 61–66. doi:10.1287/inte.20.5.61

Coon, A. C. (1997). Spatial correlation of detections for impulsive echo ranging sonar. *Johns Hopkins APL Technical Digest*, *18*(1), 105–112.

Cox, H. (1989). Fundamentals of bistatic active sonar. In Y. T. Chan (Ed.), *Underwater acoustic data processing* (pp. 3–24). Springer Netherlands. doi:10.1007/978-94-009-2289-1_1

Craparo, E. M., How, J. P., & Modiano, E. (2011). Throughput optimization in mobile backbone networks. *Mobile Computing, IEEE Transactions on*, *10*(4), 560–572. doi:10.1109/TMC.2010.187

Craparo, E. M., & Karataş, M. (2014). *Sensor optimization in multistatic underwater sensor networks.* Monterey, CA. (Unpublished manuscript)

De Berg, M., Van Kreveld, M., Overmars, M., & Schwarzkopf, O. C. (2000). *Computational geometry*. Springer. doi:10.1007/978-3-662-04245-8_1

Fewell, M. P., & Ozols, S. (2011, June). *Simple detection-performance analysis of multistatic sonar for anti-submarine warfare* (Tech. Rep. No. DSTO-TR-2562). Edinburgh, South Australia: Defence Science and Technology Organisation.

Fewell, M. P., Ozols, S., & Rzetelski, P. K. (2011). *Remote multistatic receiver field as a surveillance barrier in anti-submarine warfare* (Tech. Rep. No. DSTO-TR-2573). Edinburgh, South Australia: Defence Science and Technology Organisation, CLASSIFIED.

Freeman, H., & Shapira, R. (1975, July). Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM*, *18*(7), 409–413. doi:10.1145/360881.360919

Gong, X., Zhang, J., Cochran, D., & Xing, K. (2013). Barrier coverage in bistatic radar sensor networks: Cassini oval sensing and optimal placement. In *Proceedings of the Fourteenth ACM International Symposium on Mobile ad hoc Networking and Computing* (pp. 49–58).

Graham, R. L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information processing letters*, *1*(4), 132–133.

Klee, V., & Minty, G. J. (1972). How good is the simplex method. *Inequalities*, *III*, 159–175.

Leon, S. J. (2010). *Linear algebra with applications* (8th ed.). Upper Saddle River, NJ: Prentice Hall.

Ozols, S., & Fewell, M. P. (2011, June). *On the design of multistatic sonobuoy fields for area search* (Tech. Rep. No. DSTO-TR-2563). Edinburgh, South Australia: Defence Science and Technology Organisation.

Simakov, S. (2008, July). Localization in airborne multistatic sonars. *IEEE Journal of Oceanic Engineering*, *33*(3), 278–288. doi:10.1109/JOE.2008.927916

USN. (2014, March). Other procurement, Navy. In *Department of Defense fiscal year (FY) 2015 budget estimates* (Vol. 3).

Washburn, A. R. (2010, August). *A multistatic sonobuoy theory* (Tech. Rep. No. NPS-OR-10-005). Monterey, CA: Naval Postgraduate School.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California