RADGRAD: CREATING SOFTWARE TO ACADEMICALLY, PROFESSIONALLY, AND SOCIALLY OPTIMIZE THE UNDERGRADUATE EXPERIENCE

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

AUGUST 2017

By

Amy M. Takayesu

Thesis Committee:

Philip M Johnson, Chairperson

Keywords: Software Development, Educational Technology, Social Network, Degree Planner, Gamification

Coming soon!

ACKNOWLEDGMENTS

Coming soon!

ABSTRACT

Coming soon!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

There are a lot of things that can make a department like no other–exceptional people, exceptional facilities, exceptional events, or exceptional work. I may be biased, but I believe the ICS department at UH Manoa is a department like no other–and is exceptional in all these ways. The people are exceptional–I came into this major not knowing what to expect, but the people I have met along the way have exceeded all of my expectations. I have met some of the most brilliant, the most passionate, the most interesting and the most genuine people in this department. The facilities are exceptional–the computer labs are well maintained, and the new ICSpace is a great place for members of the department to socialize and create a sense of community. The events are exceptional–within this major, there are so many opportunities to participate in events. There are workshops, hackathons, competitions, and open source projects just waiting for participants. Finally, the work itself is exceptional–again, I may be biased, but I have grown to become very passionate about computer science, and I have met many others in the department who feel the same way. Computer science is constantly changing and it is a challenge to keep up with it, but it definitely keeps things interesting.

But nothing is perfect. Data gathered from ICS students from 2008 to 2016 on the Hawaii technology community site, TechHui, suggests that the following ten categories have constantly displeased students over the past 8 years:

1. The ICS department needs to offer classes more frequently.
2. The ICS department needs to offer a wider variety of classes.
3. The ICS department needs a better sense of community.
4. Some of the professors in the ICS department need improvement teaching.
5. The ICS department should offer more focused areas of study.
6. ICS classes are too time consuming and take up more time than anticipated.
7. The ICS department should offer more classes that meet focus requirements.
8. ICS books are too expensive.
9. Classes should be offered more frequently.
10. ICS courses should involve more group work and there should be more interaction encouraged among students in general.

There were also some other complaints among students on TechHui that werent as common but stuck out to me nonetheless. There were at least eight students who mentioned that they felt intimidated when they started out in ICS, due to the impressions they got from their classmates and the major overall. This discouraged them in several ways and had an overall negative impact on their ICS experience. As ideal as it would be, it is hard to meet the needs of all current, past and present students in a department. However, after taking student feedback into consideration, several of these problems could potentially be alleviated by creating an online platform (what could computer science students want more?) which provides students with the help they need–academically, professionally, and socially. By combining three aspects (degree planner, social network, and gamification), a new concept called RadGrad could change the department completely.

CHAPTER 2
RELATED WORK

RadGrad can be reduced into three major parts: degree planner, social network, and gamification. All three of these parts are combined to create a robust, interactive, and effective system to enhance the academic journeys of current and future ICS students.

2.1 Degree Planners

2.1.1 STAR

STAR is the degree planning system currently used by the University of Hawaii. As of September 2016, the student interface provides five main capabilities: Academic Essentials, Graduation Pathway, What If Journey, Transcripts, and Scholarships.

2.1.1.1 Academic Essentials

This interface provides information about the student's academic progress, and compares it to the student's academic requirements to show how close the student currently is to graduation. This information includes credit totals, grades, and required courses. This interface also includes a section for "Advisor Notes", which is filled out during advising sessions. There is another section for "Events and Actions" which lists important student academic events such as college applications, admittance, and graduation, and student academic actions such as Deans List award. A third section is called "Educational Goals", which provides the student's "immediate goals" and "highest ed goals" on a semester-by-semester basis. This information is provided by the student through occasional assessments upon log-in to STAR. The top of the page also has a section for students with financial aid.

2.1.1.2 Graduation Pathway

This interface is provided for certain programs or exploratory or pre-major students. It shows the course information for the courses that the student has taken previously and is currently enrolled in, and shows which requirements each course fulfills. It also shows future semesters and suggests future types of classes that the student should enroll in, in order to fulfill their major requirements. This interface does not suggest specific classes, but only lists the requirement that the class will need to fulfill.

2.1.1.3 What If Journey

This interface is provided for undergraduates at UH Manoa. It allows students to choose a different major than the one they are currently in. The page then reloads to show the STAR homepage, altered to show the requirements of the chosen major. This shows students where they would be in the program if they were to switch majors.

2.1.1.4 Transcripts

This interface allows students to access their campus transcripts by semester and by department. It also allows transfer students to access their transfer transcripts by semester and by institution.

2.1.1.5 Scholarships

This interface allows students to find scholarships by either using a keyword search or by selecting the "My Best Fit Scholarship" tab, which presumably gathers student academic data and compares it with scholarship data to find matches.

2.1.2 MyEdu

The MyEdu website states that it aims to help students succeed in college, tell their stories, and get jobs and internships. It is free to sign up, and the website reports that "twice as many MyEdu users graduate on time, compared to the national average." There are seven main capabilities of this website: GPA Calculator, Schedule Planner, Profiles, Internships/Jobs, Professor Recommendations, Course Grades, and Degree Timeline.

2.1.2.1 GPA Calculator

Students can input grades and course information to calculate their course GPA, semester GPA, and overall GPA. The calculator allows students to define grading styles and scales for each course, and set grade totals as well, which can assist students in figuring out minimum grades needed for desired final grades. The GPA calculator also comes with an Assignment Tracker, which allows students to input their assignments, due dates, and outcomes. This feature can also email automatic reminders to students regarding due dates. Finally, there is also a questions & answers section which allows students to interact with other students in the same major.

2.1.2.2 Schedule Planner

Students can create class schedules based on professor recommendations, grade distribution, and the time of the day. Schedules can be easily shared among classmates and study groups. There are also class schedule generator features, which includes an auto scheduler that automatically plans a student's schedule given professor recommendations, overall GPA, preferred days of the week, and preferred times of day. When testing out this feature, not all ICS classes were listed, and I was only allowed to choose the ones listed. After submitted one class, I got an nondescript error and was not allowed to proceed.

2.1.2.3 Profiles

Students can create a personalized profile which showcases accomplishments, projects, volunteerships, internships, and work experiences. These profiles are public to other members, encouraging students to make their profile stand out from the crowd. Student information is displayed in a Windows 8-esque type of block format.

2.1.2.4 Internships/Jobs

MyEdu includes a job search engine boasting "over 2,000,000 entry level job listings." Users can search through the database using location, keywords, and job type. Users can save jobs to review later and save jobs they plan to apply for to keep their job search organized.

2.1.2.5 Professor Recommendations

Professor recommendations are integrated within other MyEdu applications. These recommendations include study tips and exam types, lecture and attendance policy, teaching style and effectiveness, and official school evaluations. MyEdu prides their recommendation system on being superior to other popular recommendation systems (i.e. RateMyProfessor) due to including more relevant rating categories, as opposed to purportedly less relevant categories such as "hotness".

2.1.2.6 Course Grades

MyEdu works with universities to incorporate official course grade records. However, instead of just showing the students their personal records, MyEdu takes it up a notch by also displaying all encompassing statistics, such as class graduate distribution, average course GPAs, and average drop rates. These types of data can assist students in making more educated decisions in the future when they sign up for courses. Similar to my experience with the Schedule Planner, when testing out this feature, not all ICS classes were listed, and I was only allowed to choose the ones listed. After submitted one class, I got a server error and was not allowed to proceed.

2.1.2.7 Degree Timeline

This feature shows an overall view of the student's progress in graphical form. Similar to STAR's academic essentials, it displays course totals, credit totals, and graduation progress percentages. MyEdu stresses that this feature allows students to avoid the "senior surprise."

2.1.3 Starfish by Hobsons

The slogan for Hobsons is “Education Advances: Imagine a world where all students find their best fit.” Hobsons offers a wide range of educational solutions, ranging from students K-12 to college students. Starfish by Hobsons is one of their platforms which focuses on success, support, and retention initiatives, and engaging students more effectively with the campus community. There are three main parts of the Starfish Enterprise Success System: Early Alert, Connect, and Degree Planner.

2.1.3.1 Early Alert

Early Alert is a early warning and student tracking model which mines student performance data from existing technologies at the particular institution to detect at-risk students. These students are detected early enough, such as at the first sign of a problem, so that there is enough time to make a difference. There is a type of reward system called Kudo (a positive feedback note), which is used to encourage students and reward them for improvement or good work.

2.1.3.2 Connect

Connect is an online appointment scheduling and case management system. This system promotes communication between students and their advisers, instructors, and tutors by means of in person meetings, phone calls, or virtual meetings. Connect includes a kiosk to allow easily scheduled walk-in meetings. These kiosks can help staff to manage a student queue and also allows students to check wait times remotely, which can save a lot of time and frustration. Connect also includes a road map for each student, which documents the steps a student must take to achieve his or her goals. This map is created by an adviser and is visible to all members of the student’s support network.

2.1.3.3 Degree Planner

Degree Planner provides academic templates which advisers can use to easily edit to adjust to a particular student’s needs. It also focuses on students’ constantly changing goals and ability to adjust the student’s plan to accommodate these goals. When a student deviates from their given plan, the student’s adviser is notified so that they can plan a meeting with the student to check on their status and re-identify their goals.

2.1.4 College Scheduler

The College Scheduler company has two products: Schedule Planner and Pathway Planner. The Schedule Planner focuses on optimizing the way students can plan their schedules, and the Pathway Planner focuses on optimizing the way students progress towards graduation.

2.1.4.1 Schedule Planner

Schedule Planner allows students to easily schedule (or automatically generate) their classes around outside obligations. It also helps students to maximize their credit hours and graduate on time. Schedule Planner also analyzes student preference data to predict the optimal number of course sections to offer and helps to evenly distribute class fill rates. It enables advisers to create course schedules for groups of students at a time. One of their main goals is to allow students to focus on which courses to take rather than worrying about when they are being offered.

2.1.4.2 Pathway Planner

The Pathway Planner allows students to plan their schedules in a multi-year format to encourage seeing the bigger picture and to plan ahead. It provides visuals to show students how their predicted course loads will affect their graduation date. Administrators can also see the courses that students plan on taking before registration. This allows for the addition and elimination of courses to best fit student needs.

2.1.5 Coursicle

The slogan of Coursicle is "Course registration sucks but Coursicle makes it better." The features of Coursicle are: students can receive text or email notifications when a seat opens up in class, students can schedule their courses using an attractive schedule planner, students can search through courses more easily with a variety of filters, students can create schedules with all prospective classes and then narrow them down to one workable schedule, students can easily compare textbook prices online through Coursicle, and students can view what classes their classmates are signed up for via Facebook.

2.1.6 Individual Student Software

There are other types of download-able software currently available for students to use individually. These systems are for individual use, and are not tailored for institutional implementation. To use these systems, students input information about their education, such as classes, credits, and requirements. This data is then used to create organized visualizations to help students to better see their goals and pathway. A popular generic system is the Microsoft Office College Credit Planner Template. Many individual colleges and universities have their own custom download-able course planning spreadsheets as well.

2.2 Social Network

2.2.1 LinkedIn

LinkedIn is widely known for being the world's largest professional network. It sets itself apart from other popular social media sites by being focused solely on building professional identities and forging professional relationships. There are six major components to LinkedIn: Home, Profile, My Network, Learning, Jobs, and Interests.

2.2.1.1 Home

A user's homepage is arranged in a feed type format, with quick information about your profile, profile views, and incoming messages. The feed section contains recent updates from connections and companies related to your interests. There are also sections that encourage engagement–for instance, quick ways to "share an update", "upload a photo", or "write and article" and suggestions to "reconnect with your colleagues" and to add someone you may know.

2.2.1.2 Profile

A user's profile page is available for other LinkedIn users to see. Users can decide what information they would like to share about themselves, but it is all limited to professional related categories such as education, work experience, volunteer work, and skills and endorsements.

2.2.1.3 My Network

A user's network includes current connections, recommended connections, connections added through outside contact information, and contacts added through an alumni network.

2.2.1.4 Learning

LinkedIn offers online courses on professional development topics such as leadership, storytelling, creating alliances with employees, and winning back a lost customer. There are also field-related courses, such as online code courses. These courses are often in the form of videos, and can be accessed by premium LinkedIn members.

2.2.1.5 Jobs

Jobs on LinkedIn automatically suggest jobs for users based off the information on their profile. Jobs can also be searched for using keywords such as job title, company, and location. Users can set preferences to refine their automatic suggestions.

2.2.1.6 Interests

In the Interests section, users can follow companies and groups based off their personal interests. There are also links to SlideShare and ProFinder, which offer services for creating professional presentations and hiring local freelancers, respectively.

2.2.2 TechHui

The TechHui page describes itself as being "Hawaii's Technology Community." The TechHui site has ten main sections: Profile, Members, Events, Forum, Groups, Photos, Videos, Blogs, Directory, and Coders.

2.2.2.1 Profile

Each user has a profile page which contains information such as a name, profile picture, occupation, areas of interest, software language proficiency and interests, and recent activity.

2.2.2.2 Members

The members page lists all members, including a section at the top for featured members. Each member is listed by their name, with their profile picture and location. Through this page, users can communicate with other users by commenting on other user's profile pages.

2.2.2.3 Events

The events page lists upcoming events and featured events. The event snippets include an imagine, a name, a time and date, a location, the name of the organizer, the type of event, and a brief description of the event. Users can click on these snippets to go to an event page, which includes more detailed information and allows users to respond to events with "will attend", "might attend" and "will not attend."

2.2.2.4 Forum

The forum page includes a list of technology related categories, which can be clicked on to access a list of related forums. It also includes some featured forums at the top. Some of these categories include "General Software Development", "Java Software Development", "Funding Technology Startups", "Software Design Patterns", "Tech Jobs", "Tech Resumes", "Web Design", "Tech Humor" and more. Users can both start discussion forums and respond to other users' forums.

2.2.2.5 Groups

There are many different groups listed on this page, including some featured groups. Each group snippet has an image, a name, the amount of members in the group, the date of the group's latest activity, and a brief description of the group. Users can click on these snippets to learn more about the group and to join the group as well. Once in the group, users can participate in commenting on the group wall and creating and responding to group discussion forums.

2.2.2.6 Photos

On the Photos page, users can easily view all public photos uploaded by users (including profile pictures). Featured photos are included as well. Users can view these photos and comment on them as well.

2.2.2.7 Videos

On the Videos page, users can easily view all public videos uploaded by users. Featured videos are included as well. Users can view these videos and comment on them as well.

2.2.2.8 Blogs

This page displays a feed of all users' blog posts. Posts are also organized by featured posts, latest posts, most popular posts, and monthly archives. Users can click on these blog posts to read them in their entirety and can comment on them as well.

2.2.2.9 Directory

This page includes a listing of technology related jobs in Hawaii, organized into 21 subcategories. Users can click on these listings to view more details about the jobs, and also to access external websites.

2.2.2.10 Coders

This page lists web startups that are writing code in Hawaii. The list contains just the names of the startups, which can be clicked on to learn more at the startup website.

2.2.3 Rate My Professors

Rate My Professors allows users to communicate and share content with each other by posting reviews of colleges and professors. Although users can create accounts, the reviews are listed as anonymous. Other users can provide feedback on reviews with either a thumbs up (user found this to be useful) or thumbs down (user did not find this to be useful). The site also contains site-generated blog posts and videos, but users cannot directly interact with these.

2.2.4 Other Popular Social Networks

Social networks have become extremely popular and there are too many of them to describe in detail here. The top fifteen most popular social networks as of September 2016 are Facebook, Instagram, YouTube, Twitter, LinkedIn, Pinterest, Google+, Tumblr, Reddit, VK, Flickr, Vine, Meetup, Ask.fm, and ClassMates. While most of these are not academically focused, they could potentially host an academic environment. Additionally, while RadGrad could be integrated directly into one of these existing social networks (i.e. become a Facebook application), creating a standalone application does not exclude members who do not have a Facebook or are not active on Facebook, it does not depend on the continuing popularity of Facebook, and I believe it may develop a stronger sense of brand.

2.3 Gamification

Since it would be ineffective and senseless to discuss every existing video game, I conducted a brief informal survey of some current ICS undergraduates regarding their current favorite video game. They listed the following games: League of Legends, Monster Hunter, sports games (i.e. NBA2k7), Hearthstone, RimWorld, Geometry Dash, Overwatch and Kerbal Space Program. In this section I will discuss my current favorite video game and three of the popular video games according to the ICS students.

2.3.1 Summoners War

Summoners War is a mobile fantasy RPG with over 60 million players worldwide. It is based off a freemium model, with many players playing for free, and many other players playing with in-application purchases. Based off the iTunes Summoners War page, the basic premise of Summoners War is as follows: “Jump into the Sky Arena, a world under battle over the vital resource: Mana Crystals! Summon over 900 different types of monsters to compete for victory in the Sky Arena! Assemble the greatest team of monsters for strategic victories!” As with many games, one of the interesting things about it is it’s ability to motivate users into completing several, often tedious and unenjoyable actions, in order to achieve a virtual reward. One of the examples of this is the weekly Arena Rank. The Arena is where players can battle against other players, in an attempt to reach as high a rank as possible. There are different ranks based off the amount of victories and defeats the player has had: Beginner, Challenger, Fighter, Conqueror, Guardian, and Legend. The Arena is often very difficult, and in order to achieve a certain standing, the player must constantly battle others, and set up a solid defense that cannot be defeated by other players. In order to improve one's defense and offense team, a player must spend hours doing grueling tasks, such as gathering magical essences, gaining EXP points to level monsters, and collect "runes" which can be strategically placed on each monster to improve certain stats. Doing these tasks can take up a significant amount of time and energy. Each week, a player is awarded a certain rank based off their performance in the Arena that previous week. This rank is manifested in the form of a small icon next to the player's name. When players see other players' icons, they are immediately informed of that person's standing in the game, and the player him/herself will be rewarded with feelings of pride and satisfaction.

2.3.2 League Of Legends

League of Legends is a multiplayer online battle arena (MOBA) type of video game and also follows a freemium business model. In this game, the player assumes the character of a summoner who controls a champion with unique abilities, and they battle with a team of other champions against another team of champions (either other live players or computer controlled). The main goal of the game is to destroy the opposing team's nexus, which is a structure at the middle of the team's base and is protected by defensive structures. At the start of each match, all champions start off weak, but they can increase in strength throughout the game by accumulating items and experience. Each match typically lasts from 20-60 minutes. There are three different game modes: Summoner's Rift, Twisted Treeline, and Howling Abyss. Each game mode is similar in that a team of players must work together to accomplish a terminal objective and a victory condition. Each mode also includes smaller intermediate objectives that can help teams to get closer to victory. Opposed to Summoner's War, gold gathered during the match and items purchased with that gold only last for that match, and do not carry over to future matches. Each match begins with each player being more or less equal in terms of advantage, regardless of how much time or effort the player has put in beforehand. However, the game does include other incentives to continue to win games and see personal development. Players get player experiences from playing matches on a single account. As their experience increases, they can ascend from level 1 to 30. Higher level players are given access to different maps, game modes, and additional abilities and features which give players a small boost in battle.

2.3.3 Hearthstone

Hearthstone is a free to play online collectible card video game. It is turn based between two opponents, who use constructed decks of thirty cards, and a selected hero with a unique power. Players can attack the opponent using mana points. The main goal is to reduce the opponent's health to zero. If the player wins, they can earn in-game gold, new cards, or other in-game prizes. Players can use the gold or microtransactions to purchase new cards to improve their decks. There are several different game modes: casual and ranked matches, daily quests, and weekly challenges.

Unlike many other popular collectible card games, Hearthstone does not allow players to trade cards. Instead, players can disenchant their unwanted cards into arcane dust, which can then be used to craft new cards of the player's choice.

2.3.4 Overwatch

Overwatch is a team based multiplayer first person shooter (FPS). Each team has six players, and each player may select one predefined hero character. There are four classes of heroes: Offense, Defense, Tank, and Support. Each hero character has unique movements, attributes, and skills. As the team is being set up, the game will provide advice if the team is unbalanced. However, once the game starts, players can still switch characters after a death or after returning to their home base. The team of heroes work together to secure and defend certain control points and/or escort a payload across the map in a certain amount of point. As players continue to play matches, they can gain rewards that do not affect gameplay, such as character skins and poses. At the end of each match, a server-determined Play of the Game (PotG) is replayed for all players. This play is based off certain factors such as a high scoring moves or effective use of a skill. Up to four individual achievements per team are highlighted, and afterwards players can vote for one to promote. The player who wins the most votes get a reward of experience points. As players gain experience points, they can earn a loot box, which provides certain in-game prizes and in-game currency. If players do not have enough experience points for a loot box, they also have the option to obtain one through a microtransaction. The game supports several different gameplays such as tutorial and practice modes, casual matchmaking, weekly brawls, custom games, and competitive play. Casual matchmaking allows players to play alone or with friends, and are randomly matched against others with similar skill levels. The weekly brawl gameplay was inspired by Hearthstone, and features matches with unique rules, which change weekly. Custom games allow users to have private or public games and can edit different options for that specific match. Competitive mode allows players within a certain region and on a certain platform, to become ranked. This mode is run in 2.5 month seasons. Only players at level 25 or above can participate. Participants also much first play ten preliminary matches which will assign the player a skill rating from 1 to 5000, which is used to create ideal matches. Similarly to the arena battles in Summoners War, there are seven skill ranking tiers: Bronze, Silver, Gold, Platinum, Diamond, Master, and Grandmaster. Players can be demoted to a lower tier or promoted to a higher tier based on their performance. Each competitive win awards a player with in-game currency. Players will also get an additional award based on their final ranking at the end of the season.

CHAPTER 3
DESIGN

3.1 Program Requirements

3.1.1 Motivation

While detriments to SAON technologies are well documented [2] [8] [9] [11] [19], there few tools/services to analytically design SAONs around them. Furthermore, none of these tools/services are free and open-source.

3.1.1.1 Cost Efficiency

Section **??** discusses the costs of marine telemetry systems, noting that acoustic telemetry systems produce data at a significantly lower ($\geq$10x cheaper) cost than VHF, or GPS, or other Satellite based technologies. In order to maintain the cost-efficiency of acoustic technology, a significant number of detections $\geq 10\%$ of the produced transmissions must be captured by the SAON's receiver array. Given the numerous (but avoidable) impediments to reception of these acoustic signals (**??**), the array-design process becomes critical to maintaining the cost-efficiency of SAON technologies. A free network design tool would help to maintain the cost-efficiency of SAONs by eliminating costs surrounding their design and evaluation, and increasing their data recovery rates.

3.1.1.2 Metrics

The computation of network metrics (Absolutes Recovery Rate, Unique Recovery Rate, Network Sparsity) is very labor intensive at large scale. Additionally, the process of computation may vary from experiment to experiment. An automated tool would solve both issues by providing a fast, repeatable, reliable, and well-documented method for computation. Metrics from such a tool would be useful in directly comparing different network designs.

3.1.1.3 Transparency

An open-sourced tool/service would make the design process more transparent, permitting peer-review and modification. This would provide increased confidence in the process, and increased adoption of the tool. Increased adoption would result in a larger number of efficient SAONs, leading to higher data recovery rates, better data quality, increased return-on-investment, and the ability to better address research questions.

3.1.2 Supported Workflows

3.1.2.1 Static Analysis

As mentioned in section 3.1.1.2, a primary motive for this tool was the ability to create a repeatable means of measuring the performance of a SAON. To this end, the ability to measure an existing network design is important. Users should be presented with network metrics after specifying bathymetry, receiver locations, network properties, and an animal model for a given study site.

3.1.2.2 Optimal Design

The primary motive for this tool is the ability to design optimal SAONs. Users should be presented with a network design (optimal receiver locations), and network metrics after specifying bathymetry, the number of receivers in the network, network properties, and an animal model for a given study site.

3.1.2.3 Optimal Addition

Similar to the problem of optimal design, is the problem of optimal addition: the augmentation of an already existing SAON. Users should be presented with a network design (optimally augmenting receiver locations), and network metrics after specifying bathymetry, the number of receivers to add to the network, network properties, existing receiver locations, and an animal model for a given study site.

3.2 Conceptual Model

3.2.1 Time/Space Modeling

3.2.1.1 Spatial Modeling

To model various attributes of a 3-dimensional underwater environment, several two-dimensional Grid of cells (in the x and y dimensions) containing numerical values are used. Numerical values in those cells, combined with user-defined values, can then be used in various shape functions to generate a third dimension (z) of values for that cell. In this way, a significant amount of memory is saved by computing values for a specific three-dimensional cell as needed, instead of storing an additional dimension of values.

3.2.1.2 Temporal Modeling

Modeling a 3-dimensional environment over time is computationally intensive, as individual values can vary with time. The primary temporally-dependent phenomena considered by this system is that of animal movement. The animal model does not represent individual animal movement at a specific time, but the summary cell-residency of all tagged individuals over the entire study period. As a result, temporally-dependent phenomena need not be considered.

3.2.2 Bathymetric Modeling

3.2.2.1 Bathymetric Grid

Bathymetry files are generally given as two-dimensional matrix of numerical values or a list of x,y,z values. Bathymetric files describe a three-dimensional space as a regular grid of rectangular prisms (cells) with constant length (x-dimension) and width (y-dimension), but varying negative (depth is negative) heights (z-dimension). The resolution of a bathymetric file is given by the x and y (length and width) dimensions of its cells. For example, a 50 meter Bathymetry file has cell sizes of approximately 50 meters square (although these cells are not necessarily perfectly square). Bathymetric files include meta-data listing the file's beginning and ending coordinates (North/South Latitudes and East/West Longitudes), the grid size (the number of rows and columns) in cells, and grid resolution (either in linear distance or degrees of latitude/longitude).

MANDe works on a two-dimensional, grid-based system, taking advantage of the grid given by the user-provided bathymetric file. The Bathymetric Grid is a 2D Grid containing numerical values that describe a third dimension (depth). The exact spatial extent and resolution of this description are given by the bathymetric file. Thus, the resolution of MANDe's simulation and output are dictated by the resolution of the input bathymetric file.

A cell x (with row index i and column index j) on the Bathymetry Grid (A) is referred to as A_x or $A_{(i,j)}$.

3.2.2.2 Bathymetric Filetypes

Two highly popular file formats for Geographical Information Systems are provided by NetCDF and ArcGIS. NetCDF provides an open source file format that lists a header of meta-data and a white-space delimited matrix of numerical values. ArcGIS is a private institution that supplies many different file types, formats, and encodings for a family of GIS-related software systems. ArcGIS also supports the encoding and transcription of its proprietary formats to the NetCDF format. Due to the large number of possible format and encoding combinations in ArcGIS file formats and the ability to translate file these various formats to NetCDF, MANDe supports the NetCDF standard by default.

3.2.2.3 Bathymetric Resolution

Two key components of this system are the animal model and the bathymetric shadowing model. These models make decisions based upon the depth at a particular cell and the distance between cells, data which is governed by the input bathymetry file. As stated in Section 3.2.2.1, the resolution of the program's output is dependent upon the resolution of the input bathymetry file. Obviously, higher resolution grids will offer higher resolution results; but, high-resolution bathymetric files tend to be difficult to come by. These files are often held by private agencies or simply never released to the public. It may then seem useful to artificially increase the resolution of the simulation by dividing the input file's bathymetric cells into sub-cells of finer resolution, but doing so increases the computational size of the program without meaningfully increasing the accuracy of the results.

When subdividing cells, either the sub-cells are given the same depth as their parent cell, or the depth of a sub-cell is interpolated from surrounding cells by some smoothing function. Subdividing a cell into sub-cells with the same depth makes the assumption that all sub-cells are actually the same depth. Furthermore, this results in the animal and bathymetric shadowing models making the same depth-based decision for all sub-cells that they would for the larger parent cell, needlessly increasing the computational load (see Figure 3.1b). Subdividing a cell into sub-cells with a depth governed by a smoothing function makes the assumption that there are no impeding obstacles between neighboring cells, and that there is a smooth transition between them (see figure 3.1c). Subdividing the two cells in Figure 3.1a half (ignoring for now the y component of the grid) results in the four sub-cells with depths given by a smoothing function (Figure 3.1c) would result in the large depth change at the sheer cliff face in Figure 3.1a being smoothed into smaller changes in depth, which would allow the unobstructed transmission of acoustic signals.

Both strategies (duplicating depth and applying a smoothing function) for artificially increasing the resolution of a bathymetric file disregard the manner in which the bathymetry was originally observed. Bathymetry is almost always computed as the average observed depth at several points within a geographic area of a given size (resolution). For example, imagine a particular cell in a bathymetric grid has a steep cliff running across the middle. Assuming that the sea floor at the top and base of the cliff were perfectly flat, and one measurement was taken at the top of the cliff and one at the bottom, the cell would have a depth equal to the average of the two observed depths. This average depth would then represent the depth for that entire cell, modeling it as a perfectly flat surface. Obviously this is problematic as the true nature of the sea floor is misrepresented. This misrepresentation leads to two conflicting arguments, the first is that the application of smoothing functions or duplicating depths of already averaged data makes faulty assumptions about real-world bathymetry. On the other hand, because source bathymetric files already represent aggregate data, one could argue that any conclusions drawn from the source bathymetry files are already faulty. Therefore, the conclusions one can draw are only as good as the bathymetric information available. As the resolution of measured bathymetry (bathymetric measurements taken from the real world) increases (becomes finer), so too does the accuracy of the simulation. While artificially increasing the resolution of a source bathymetric file may skew results, it is sometimes useful for meshing two bathymetric source files of varying resolution into one larger bathymetric dataset. This program does not currently support modifying the resolution of source bathymetric files.

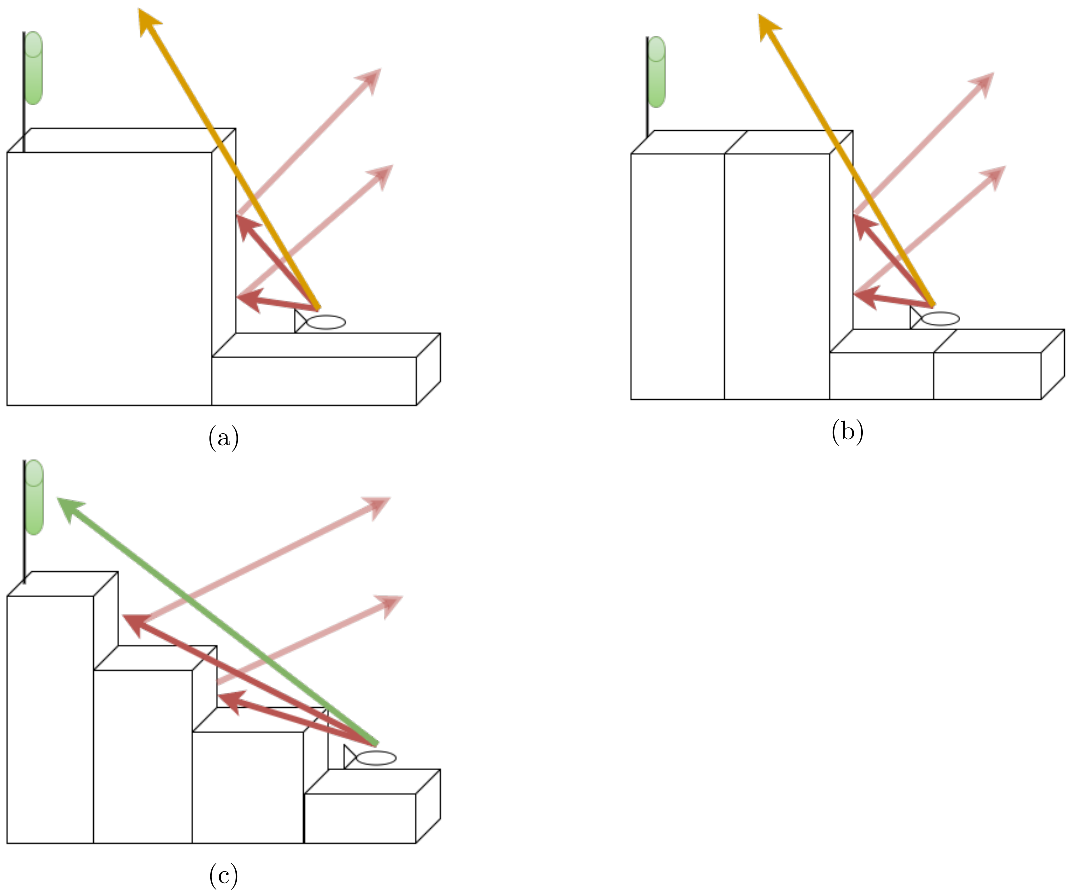


Figure 3.1: (a) illustrates the bathymetric shadowing model for two adjacent cells within a bathymetric grid. (b) shows how artificially increasing the resolution of the bathymetric grid from (a) using the duplication method of cell sub-division does not affect the bathymetric shadowing model but increases the number of cells to consider. (c) shows how artificially increasing the resolution using a smoothing function increased the number of cells to consider and can lead to inflated signal reception.

3.2.2.4 Bathymetric Shadowing

In the real world, the transmission of an acoustic ping originates from the tagged animal and propagates to the receiver. This propagation is governed by complex interactions with the surrounding environment (including the bottom substrate, water density, distance to the surface/sea-floor, thermoclines, the number of tagged animals in the vicinity, and ambient noise). Because it is difficult to model these phenomena without significant data on a large number of variables over a large area, the system uses a simplified propagation model relying on direct line of sight between a tagged animal and a receiver. Simply put, in order to observe a transmission, there must exist a physically-unobstructed path between a tagged animal and a receiver.

3.2.3 Animal Modeling

3.2.3.1 Behavior Grid

Animals exhibit many different movement patterns and habitat preferences (both of which can vary in three-dimensional space). This greatly affects their distribution within the study space, and thus the network configuration that should be deployed to capture their movement. To describe the distribution of transmissions released from tagged animals, a two dimensional Grid with the same dimensions and resolution as the Bathymetry Grid (Section 3.2.2.1) is used. Each cell in this "Behavior Grid" (B) contains a positive real value indicating, out of all transmissions that will be released over the entire study period, the percentage of transmissions that are expected to be released from within that cell's water column. This value can loosely be though of as the "animal-residency" of a cell. A cell x (with row index i and column index j) on the Behavior Grid (B) is referred to as B_x or $B_{(i,j)}$.

To populate this 2D grid with values, two behavioral models are provided to simulate the horizontal animal distribution. Two more models are provided to simulate the vertical distribution. *Note*: the terms "animal" and "transmission" are used interchangeably since acoustic transmissions are the metric being counted, but are emitted from the tags carried by animals.

3.2.3.2 Horizontal Movement Modeling

To simulate tagged animal movement across the horizontal x/y space (as one would expect to see on a map from above), two basic probabilistic movement models are provided: Random Walk, and Ornstein-Uhlenbeck(OU).

3.2.3.2.1 Random Walk Model The Random Walk model assumes that animals move randomly through the environment. As a result, over the entire study period, each valid grid cell (as defined by the Restricted Vertical Habitat Range (Section 3.2.3.3.2)) will see an equal amount of animal traffic. The result is that every valid cell in the grid will have the same chance of capturing an animal's acoustic transmission. It is assumed that tagged individuals will be willing and able to very briefly (in probabilistically negligible time frames) pass through inhospitable (over dry land, through impassible terrain) cells to get to other cells. This means that disjoint sections of habitat will contain an equal share of acoustic transmissions.

3.2.3.2.2 Ornstein-Uhlenbeck Model The Ornstein-Uhlenbeck(OU) model[12] supports the idea that over time, animals will tend to congregate near certain points of interest. This concept models an animal's desire to seek out and remain near a physically significant structure, a region of high food availability, breeding grounds, shelter, etc. The OU algorithm allows for the modeling of the attraction towards a focal point in the x and y directions separately. Assuming a center point at the origin of a Cartesian grid, increasing the attraction in the x-direction will bring the distribution closer to the x-axis, and decreasing it will spread the distribution away from the x-axis. The same holds true for the attraction in the y-direction/y-axis. A correlation value ($-1 \leq x \leq 1$) allows for tilting the angle of the distribution. A positive correlation value tilts the distribution clockwise, and a negative correlation value tilts it counter clockwise. Correlations of 0 (no tilt), 1 (180° tilt), and -1 (-180° tilt) will have no observable effect on the distribution.

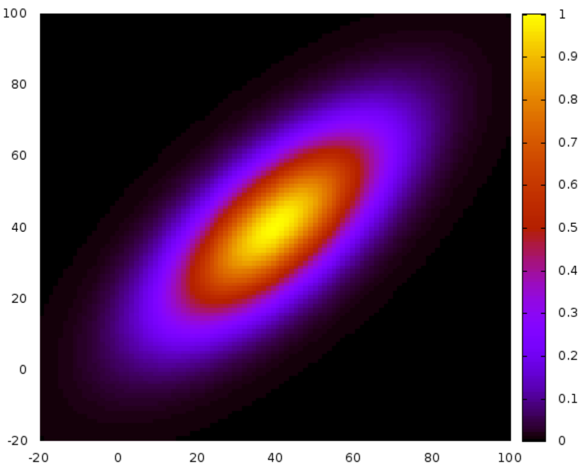


Figure 3.2: An example of a distribution given by the Ornstein-Uhlenbeck Model with a high attraction value in the x-direction, a low attraction value in the y-direction, and a correlation value of 0.7.

3.2.3.3 Vertical Movement Modeling

To model the z-axis movement behaviors of animals within the water column (as one would expect to see from the side of an aquarium), two optional movement models are provided: Habitat Preference, and Restricted Vertical Habitat Range. The first controls the distribution of animals within the water column, and the second prohibits animal residency within cells that fall outside specified depth ranges.

3.2.3.3.1 Habitat Preference Some animals exhibit the preference to reside within a specific section of the water column. For example, prey animals may prefer hiding in reef heads at the bottom of the water column, while predators will prefer to hover several meters off the bottom. This preference can be incorporated into the animal model by specifying mean (”Preferred Depth”) and standard deviation(”SD of Preferred Depth”) values. These values are given as a measure of the distance (in meters) from the bottom. For example, specifying a depth of ’0’ for Preferred

Depth indicates that the animal prefers to live on the sea floor, while a value of '5' indicates that the animal prefers to live 5m off the sea floor. Allowing a standard deviation value allows for the modeling of animals that tend to be sedentary within the water column (a small deviation), and those that migrate through the water column (a large deviation). The simulation does not support the modeling of sub-surface animals, as any part of the distribution curve falling below the sea floor will be redistributed along the rest of the curve.

3.2.3.3.2 Restricted Vertical Habitat Range Some animals will live only in a specific depth range. For example, a deep sea fish may live only in depths of 300-400 meters. To incorporate this into the behavioral model, users can specify a minimum and maximum vertical habitat range for their animal. If this option is selected, the program will only simulate animals in cells whose maximum depths are between the given minimum and maximum depths. As in the Random Walk Model, it is assumed that animals are willing and able to move between disjoint areas of habitat.

3.2.4 Receiver Modeling

3.2.4.1 Acoustic Attenuation

In practice, the highest probability of observing an acoustic transmission occurs when a tag is several dozen meters away from the receiver (due to CPDI effects [11]), and dropping off as the tag moves further away from the receiver. The attenuation of acoustic signal is modeled as a deteriorating function following a Gaussian distribution. Given a particular distance between a tag and receiver, this distribution describes the probability of capturing the transmission. This distribution is defined by user-specified values for mean value, peak value (at the mean), and standard deviation. Within this framework, CPDI effects can be modeled by modifying the mean value. Herein, the probability of detecting an unobstructed acoustic transmission from x meters away is denoted as $P_{Attenuation}(x)$.

3.2.4.2 Detection Range

The distance at which a transmission from an acoustic tag can be captured is largely determined by the transmission power of the acoustic tag. It is assumed that all animals have the same model of acoustic tag and that "transmission range" is instead a property of the receiver, referred to as "Detection Range" (D_{range}). The Detection Range of a receiver is defined to be the average distance (specific to the given study site) at which the probability of detecting an acoustic transmission drops to 5%.

3.2.4.3 Detection Area

The Detection Area of a receiver is defined as the square plane of cells around a receiver which will be considered when evaluating a potential receiver emplacement. Theoretically, the Detection Area of a receiver is a circle with a radius equal to the receiver's Detection Range (D_{range}). The Detection Area is defined as a square Grid of cells with an edge length ($d_{detection}$) equal to 2*D_{range} + 1, centered on the cell where a receiver will theoretically be placed. The additional cell ensures odd dimensions for the square, so that there exists a central cell to model as the receiver's location. A square is used instead of a circle for simplicity of book keeping (the program simply keeps track of the $d_{detection}$, and the coordinates of the Grid's top left cell). The theoretical Detection Area can then be imagined as a circle circumscribed in the square Detection Area. It might seem that this simplification would alter the probabilistic distribution models as additional "gray" cells (those cells outside the circle but within the square) are being considered. However, this is not the case as those models utilize normal functions based on the absolute distance from the receiver. Gray cells will therefore contribute exponentially less to the total probability than those cells within the Detection Range, and their effect on the final probability will be negligible.

3.2.4.4 Network Specification

There are three distinct ways to place receivers into the model: user specification, optimal placement, and optimal projection. User Specified Receivers (USR) represent receivers that are already deployed at the study site and are being integrated into a larger network. Program Placed Receivers (PPRs) are receivers that will be optimally placed by the program, taking into account existing (user specified) receivers placements using the suppression dynamic explained in section 3.4. Projected Receivers are PPRs that do not count towards the network statistic rates returned by the program. Instead, their contributed recovery rates are given separately, and represent the marginal benefits of placing incrementally more receivers.

3.3 Evaluation of Receiver Emplacements

Section 3.2.3 discusses the models used to simulate animal movement. These animal models populate the "Behavior Grid" (B), a 2D grid (of the same size as the specified Bathymetry Grid) which gives a distribution summary of transmissions across the study area, and a shape function which describes the distribution of animals within the water column at various depths. Section 3.1a discusses the Line of Sight (LoS) model used to simulate the obstruction of acoustic transmission. Section3.2.4 discusses how attenuation affects the propagation of acoustic signals and thus the probability of detecting acoustic transmissions from a particular distance ($P_{Attenuation}(x)$). Together, these models are used to evaluate receiver locations throughout the study site.

3.3.1 Goodness Grid

Recall that MANDe models a 3D environment (the study site). To avoid evaluating every cell in this 3D environment as a receiver placement, MANDe assumes that all receivers in the study will have the same elevation off the sea floor. This assumption of a fixed receiver elevation (of k meters) reduces the search space to a 2D surface parallel to the sea floor. Thus, it makes sense to store the chosen metric for evaluation of this search space in a 2D Grid with the same dimensions and resolution as the Bathymetry Grid. This Gird of metrics is referred to as the "Goodness Grid" (G). A cell x (with row index i and column index j) on the Goodness Grid (G) is referred to as G_x or $G_{(i,j)}$.

3.3.2 Evaluation Algorithms (Bias)

The Evaluation or Goodness algorithms define a process for evaluating the goodness of a particular cell as a receiver location. Given a particular cell at row i and column j, an Evaluation Algorithm will compute a non-negative, rational value representing the quantity of observed transmissions that a receiver placed k meters off the bottom of cell (i, j) would recover.

While users are able to write their own Evaluation Algorithms, three basic algorithms are provided. All Evaluation Algorithms compute the Goodness of a potential receiver location ($G_{i,j}$) by summing the number of Estimated Receivable Transmissions (ERT) from all cells within Detection Range of cell (i, j). Each Evaluation Algorithm has a particular bias in its approach to computing ERT. $ERT(x, T, a, b)$ denotes the ERT value for the cell (a,b), relative to the receiver in cell T, as determined by Evaluation Algorithm x. Note: within this paper, the terms "bias" and "algorithm" are used interchangeably when referring to Evaluation Algorithms.

Explicitly, $G_{i,j} = \Sigma_{a=i_0}^{i_n} \Sigma_{b=j_0}^{j_n} ERT(x, (i, j), a, b)$, where x is the chosen Evaluation Algorithm/Bias, i_0 and j_0 represent the top-left row/column indexes (respectively) for the Detection Area, and i_n and j_n represent bottom-right row/column indexes (respectively) for the Detection Area.
Note: here the receiver-containing cell T is denoted as (i, j).

3.3.2.1 Animal Only (Option 1)

This option prefers to place receivers in areas of high animal activity, completely oblivious to the surrounding bathymetry. This is useful for when no bathymetric information is available or transmissions are unlikely to be blocked by bathymetry (in a very flat area, or when animal activity occurs well above the sea floor). The "Animal Only" option computes ERT for a cell as the animal-residency of a cell (a, b)(according to the Behavior Grid), multiplied by the probability of detection due to attenuation for that cell's distance from the receiver cell(T).
Explicitly, $ERT(1, T, a, b) = B_{a,b} * P_{Attenuation}(Range(T, a, b))$

3.3.2.2 Visible Fish (Option 3)

This option chooses receiver locations that have the best view of areas with high animal residency. Both animal presence and visibility due to topography are considered. Figure 3.3 depicts this idea. The green and red normal curves represent the distribution of animals in the water-column. The green portion of the normal curve illustrates the portion of observable (due to bathymetry) animals.

This option is most useful when both well-documented animal behavior and high resolution bathymetry are available. ERT is computed as animal-residency (according to the Behavior Grid), multiplied by the percentage of observable transmissions (due to bathymetric obstruction), multiplied by the probability of detection due to attenuation.
Explicitly, $ERT(3, T, a, b) = B_{a,b} * P_{observation}(T, a, b) * P_{Attenuation}(Range(T, a, b))$

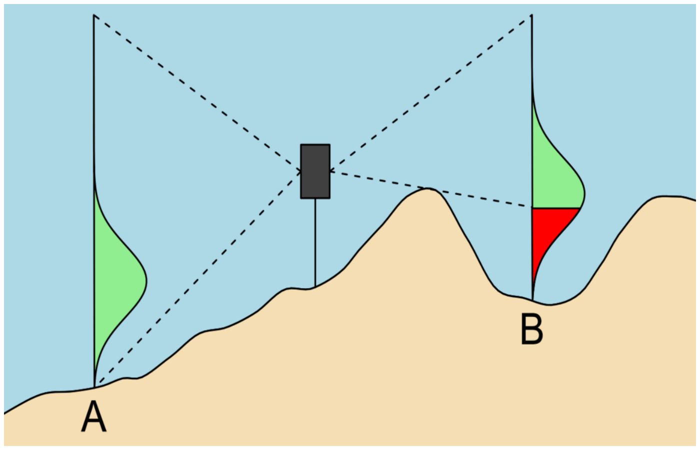


Figure 3.3: An illustration of how ray tracing and integration over a shape function can be used in Option 2 & Option 3 to compute the probability of detecting fish within a cell. Dotted lines indicate the maximum and minimum depths visible to the receiver. The normal distributions in green/red indicate the distribution of fish within a given cell as determined by a shape function. The green portion of the curve indicates the observable section of the distribution, while the red indicates the unobservable section. The observable section (green) is computed by integration over the shape function.

3.3.2.3 Topography Only (Option 2)

This option places receivers in areas that have the best visibility of the surrounding area, regardless of the expected animal-residency. This is useful for experiments where animal habitat is unknown or to be determined. Here, ERT is computed as the number of observable transmissions from a uniformly distributed animal model. While this algorithm is referred to as "Topography Only", it is implemented using the "Visible Fish/Option 3" algorithm with a simplified animal model. Specifically, it is assumed that animals are uniformly distributed throughout the environment. Because animals are uniformly distributed, the more of the water column a receiver can see, the

more transmissions it will receive. Thus, the algorithm will value cells with the best possible view of the surrounding water columns.
Explicitly, $ERT_{2,T,i,j} = B_{a,b} * P_{observation}(T,a,b) * P_{Attenuation}(Range(T,a,b))$
Note: With this option, B is initially described as a uniform distribution.

3.3.3 Line of Sight Evaluation

Section 3.2.2.4 explains the concept of Bathymetric Shadowing, which requires a bathymetrically unobstructed Line of Sight (LoS) to exist between an acoustic tag and a receiver in order for transmissions from that tag to be received. Section 3.3.2.2 discusses how the evaluation algorithms utilize this concept to compute the quantity of transmissions that are receivable (ERT). For this computation, the evaluation algorithm requires knowledge of the deepest depth ($D_{max}(p,q)$) in a cell q that is visible to a receiver in cell p. This depth is used by Evaluation Algorithms 2 and 3 to compute the finite integral for their ERT values.

To determine the D_{max} for a cell q, relative to a receiver in cell p, the cells could potentially obscure LoS between p and q must first be identified. A list of n such potentially obscuring cells is referred to as C. Figure 3.1a illustrates this process. The shaded cells in the left image potentially obstruct the vision from cell p to cell q. Recall that B_x refers to the depth of the cell x as given by the Bathymetry Grid.

Next, the slope between the receiver at p and each cell in C is determined. The slope between a receiver elevated k meters off the sea floor of cell p and v ($m_{p,v}$) is defined as the difference in elevation between $B_p + k$ and B_v divided by the Euclidean distance ($Edist_{p,v}$) between cell p and cell v. Explicitly, $m_{p,v} = \frac{(B_v - B_p + k)}{Edist_{p,v}}$. The greatest (largest signed value) slope $m_{p,v}$ for all cells v in C is the Critical Slope (m_{crit}). If a line with this slope were projected from the receiver at p to the target cell q, the line would be tangent to the tallest obstruction along that path. Thus, the critical slope determines D_{max} for cell q. Explicitly, $D_{max} = m_{crit} * Edist_{p,v} + B_p + k$. Figure 3.4 gives a visual depiction of how m_{crit} is used to determine D_{max}.

3.3.4 Selection of Optimal Emplacements

Section 3.3 describes the evaluation of cells as potential receiver locations. The Optimal Design and Optimal Addition work flows (section 3.1.2) require the identification and selection of a user-specified number of optimal receiver emplacements. Once all cells within the area of interest have been evaluated, the program selects the user-specified number of optimal receiver locations, and then the user-specified number of projected receivers.

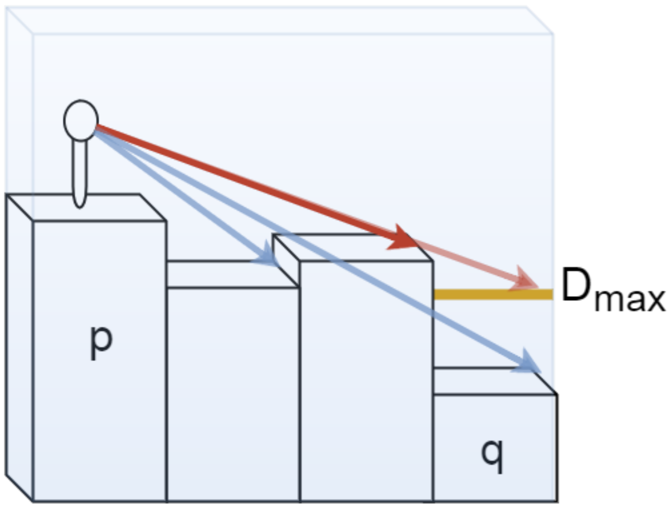


Figure 3.4: An illustration of how the critical slope, and D_{max} are computed between cells p and q. The slope between the receiver and all intervening cells are computed (blue and red arrows). Then, the greatest slope is selected as m_{crit} (dark red arrow) because it poses the greatest obstruction to vision of cell q. Finally, m_{crit} is projected into cell q (the light red arrow) to determine D_{max}.

3.4 Suppression

As previously mentioned, the optimality of a network depends greatly upon the way in which Data Quality is defined. Some users will want to design a network that covers as much of a study area as possible, while others might want to heavily saturate a small area with receivers to facilitate higher resolution telemetry. Still others may wish to find the receiver locations that return the highest number of unique data points. Unique Data Recovery Rate (UDRR) and Absolute Data Recovery Rate (ADRR) [Section **??**] also play a role in the definition of Data Quality. To allow for the control of receiver distribution, the Suppression mechanic is provided.

3.4.1 Suppression Area

As an input to the program, users can specify a Suppression Range Factor (f_{supp}) as a positive real number. This factor is used to scale the Detection Range (D_{range}) (Section 3.2.4.2) and define a Grid similar to the Detection Area (Section 3.2.4.3). The resulting Grid is referred to as the Suppression Area. The Suppression Area is centered over a receiver placement and cells within this area have their number of undetected transmissions reduced according to a Suppression Algorithm. The Suppression mechanic is used during the selection of receiver locations. After selecting each optimal receiver location (Section 3.3.4), the Suppression Area around that receiver emplacement is suppressed. As a result, receiver emplacement selection is a sequential process, where selecting an emplacement depends upon the suppression of the previous selection.

Assuming a uniform Goodness Grid, setting the Suppression Range Factor to 1.0 ensures that

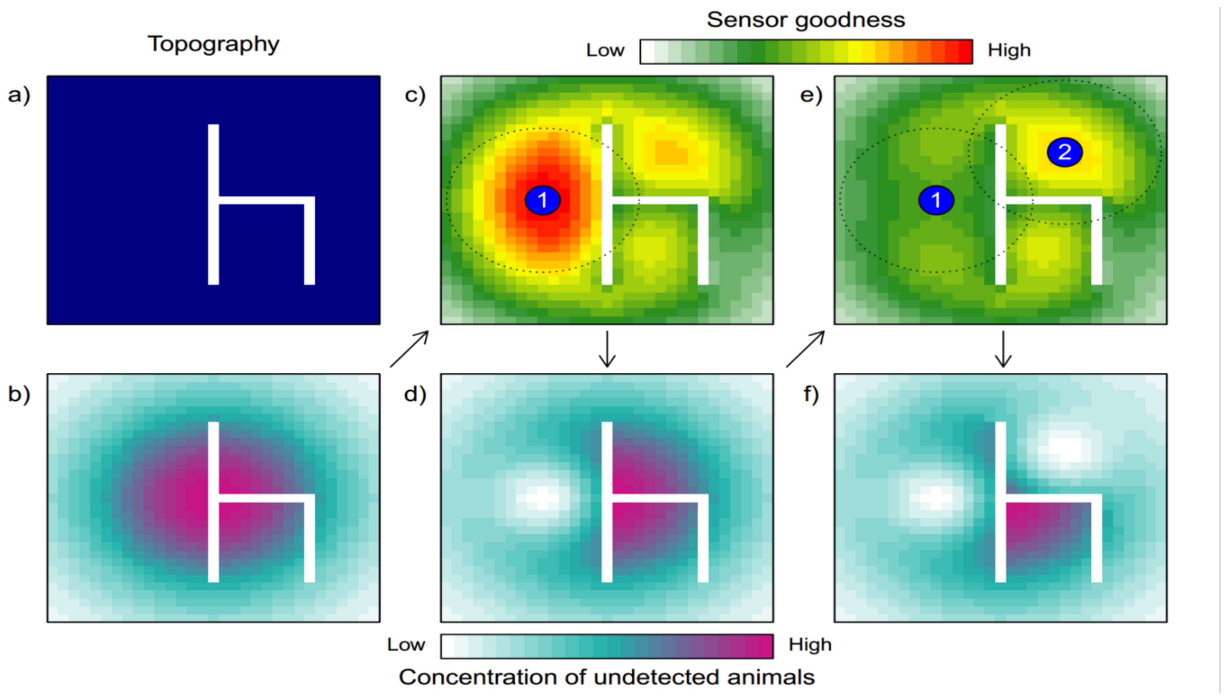


Figure 3.5: An illustration of the Exact Suppression Algorithm. (a) depicts a bathymetry grid with infinitely high walls (the white 'h' shape) on an otherwise flat plane (blue region). (b) depicts a Behavior Grid with a distribution (given by the OU movement model) of animals around the walls. (c) shows the computed Goodness of Receiver (receiver) locations within the study site. The program first identifies location 1 (the blue circle) as having the highest unique data recovery rate, and places a receiver there. The dotted lines represent the receiver's Detection Range. In (d), the program suppresses the Behavior Grid by subtracting the ERT of each cell within Suppression Range. Here the Suppression Range Factor is 1.0, so the Suppression Area is the same as the Detection Area. In (e), the Goodness Grid is recalculated, taking into account the suppressed Behavior Grid. Additionally, the program identifies location 2 as having the highest Unique Data Recovery Rate. In (f), the Behavior Grid is again suppressed to account for the placement of receiver (2).

receiver Detection Areas do not overlap, maximizing network coverage. Setting the Suppression Range Factor to 0.5 means that receiver Detection Areas will exactly overlap (the edge of one Detection Area will touch a receiver). A Suppression Range Factor of 0.0 will cause receivers to be stacked on top of each other, over the highest rated cell in the Goodness Grid. Using this approach, it is possible to disperse receivers across physical space, while optimizing the number of captured transmissions.

3.4.2 Suppression Algorithms

To promote flexibility, three algorithms for suppression are provided. The first offers a lower computation time at the cost of divergence from the theoretical model. The second offers a higher fidelity model, but requires significantly more computation. The final algorithm provides very high fidelity, but at very high cost.

3.4.2.1 Static Suppression

The Static Suppression algorithm takes a "black-out" approach and replaces the goodness of all cells within the Suppression Area by a user-specified value.

3.4.2.2 Scalar Suppression

The Scaled Suppression algorithm reduces the number of undetected transmissions in the Suppression Area using a template. This algorithm requires a minimum and maximum suppression value ($Supp_{min}$ and $Supp_{max}$). The algorithm first creates the Suppression Template, a Grid with the same dimensions the Suppression Area. This Suppression Template is populated with a radial gradient of suppression values. The central cell in the Suppression Template receives the largest suppression value ($Supp_{max}$), and the cell farthest away from the center receive the smallest suppression value ($Supp_{min}$). Cells between the central cell and the furthest cell receive a suppression value negatively correlated with their distance from the central cell (according to the Suppression Function). To suppress a particular cell, the algorithm simply preforms a scalar multiplication (corresponding cells in the Suppression Template and Suppression Area are multiplied together). Because this computation is very simple, it runs very quickly and is therefore useful for simulations that intend to place a very large number of receivers. The cost of this computational simplicity is that the algorithm ignores line of sight model. For instance, a cell whose line of sight to the receiver is obstructed will be reduced by the same factor as a cell near the receiver with an unobstructed line of sight. Obviously this is not a faithful representation of the conceptual models discussed earlier, but supplied as a computationally-simple alternative.

3.4.2.3 Exact Suppression

The primary purpose of this algorithm was to discount transmissions which will be observed, according to an Evaluation Algorithm, by a placed receiver. This algorithm uses the ERT value given by the Evaluation Algorithms (the user-specified Evaluation Algorithm is used in both Goodness and Suppression calculations) to determine the number of transmissions within a particular cell to discount. $ERT_{T,i,j}$ denotes the ERT value in cell (i,j) observed from cell T. Exact Suppression works first by computing $ERT_{T,i,j}$ for all cells (i,j) within the Suppression Area. Then, each corresponding cell in the Behavior Grid, $B_{i,j}$, is reduced by $ERT_{T,i,j}$. Finally, the Goodness of all cells within a distance of (Detection Range plus Suppression Range) cells of a suppressed cell (all those that would have their Goodness affected by the suppression) is recalculated. Figure 3.5 illustrates the Exact suppression algorithm.

3.5 Optimal Receiver Projection

In normal research situations, users will have a set number of receivers to place within their study site. The process of arriving at this number is likely unscientific, perhaps relying on user estimation. Rather than guessing at the number of receivers to use, and hoping for an adequate data recovery rate, users should be able to calculate the marginal benefit (additional detections, increased Data Recovery Rates) of utilizing a variable number of receivers. To this end, the program returns the marginal gain in Data Recovery for a given number of additional receivers. Within the program, users to specify a number of receivers to project, and receive graphs and metrics of the marginal increase in Data Recovery Rate. With this data, users can determine an appropriate number of receivers to purchase or construct an argument for purchasing more receivers.

3.6 Time and Space Complexity

To evaluate the temporal and spatial complexities of various elements of the program, the following variable inputs to the program are defined:

n	The square root of the number of cells in the Bathymetry Grid, (the edge length of a square grid). Also recall that the Bathymetry, Behavior, Goodness, and Coverage Grids are all of identical dimension.
D_{range}	The Detection Range of a receiver in the simulation..

3.6.1 Bathymetry Grid

The bathymetry data for the study site is represented as a Grid of n^2 cells. This data must be copied into local memory (RAM) from the input Bathymetry File, each cell in the grid takes O(1) time to copy, and O(1) space to hold. Thus, the creation of the Bathymetry Grid will take O(1) * n^2 = O(n^2) time and space.

3.6.2 Behavior Grid

Animal residency is computed as a function of the depth of a particular cell (Restricted Vertical Habitat), and the cell's location (OU/RW modeling). This computation takes a constant [O(1)] amount of time. The space required to store a single residency value for each of n^2 cells is also O(1). Therefore, the population of the Behavior Grid takes O(1) * n^2 = O(n^2) time and space.

3.6.3 Line of Sight Computation

As discussed in Section 3.3.3, the LoS algorithm is given as:
Step 1) Determine the m intervening cells $C_{0...m}$ between p and q
Step 2) Compute the slopes between p and C_i for all i in $[0...m]$
Step 3) Choose the Critical Slope
Step 4) Project a line from p to q along the Critical Slope to find D_{max}.

Step 1 finds intervening cells by ray tracing, which can be done in linear ($O(n)$) time. Temporarily storing this list of intervening cells requires $O(n)$ space. The number of cells considered (n), depends upon the distance between the receiver and the target cell[13]. Given that the Detection Area has a radius of D_{range}, and that receivers are located in the center of the Detection Area, the greatest distance between a receiver and a target cell in the Detection Area occurs between a receiver and the corner cells of the Detection Area (a distance of $\sqrt{2} * D_{range}$). Thus, the number of cells considered when creating a list of intervening cells is $O(\sqrt{2} * D_{range} = O(D_{range})$.

Step 2 computes the slopes for each of the m cells in C. Computing the slope between two points takes $O(1)$ time to compute and space to store. Therefore computing m slopes takes $O(1) * O(m) = O(m)$ time and space. As stated above, m is at worst $O(D_{range})$ on the Detection Area. Thus, the slope computation is at worst $O(D_{range})$, requiring $O(D_{range})$ extra storage.

Step 3 chooses the largest slope amongst all m slopes. Therefore, m items must be considered, each requiring $O(1)$ time to consider as the largest. Since, m is $O(D_{range})$ on the Detection Area, this step takes $(O(D_{range}))$ time to compute, and $O(1)$ space to store.

Step 4 is a direct computation, requiring $O(1)$ time to compute and $O(1)$ space to store the result.

In total, the LoS computation time is $O(D_{range}) + O(D_{range}) + O(D_{range}) + O(1) = O(D_{range})$, requiring $O(D_{range}) + O(D_{range}) + O(1) + O(1) = O(D_{range})$ temporary storage space.

3.6.4 Goodness Grid

Recall that the population of the Goodness Grid is performed by Evaluation Algorithms (Section 3.3.2), which compute ERT values for each of the D^2_{range} cells in the Detection Area. These ERT values are then summed (requiring $O(D^2_{range})$ time) to provide the Goodness value for a single cell, out of the n^2 cells, in the Goodness Grid. Thus, if the computational complexity of the ERT computation of Evaluation Algorithm x is defined to be E_x, the computational complexity for populating the entire Goodness Grid is the time for ERT computation across the D^2_{range} cells in the Detection Area plus the time for summing those D^2_{range} ERT values, for each of the n^2 cells in

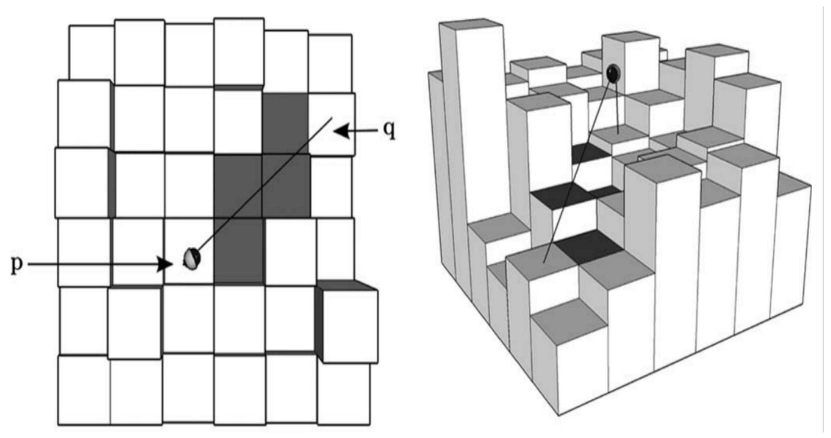


Figure 3.6: An illustration of how ray tracing is used within a 3D environment to identify potential visual obstructions. Ray tracing is used to determine which cells potentially block the line of sight between the receiver-containing cell p, and the target cell q. Shaded cells must be evaluated by the Line of Sight algorithm to determine which portion of the water column in q that is visible from p. [2]

the Goodness Grid. Explicitly, this is $O(n^2 * [D_{range}^2 * E_x + D_{range}^2]) = O(n^2 * D_{range}^2 * [E_x + 1]) = O(n^2 * D_{range}^2 * E_x)$.

3.6.4.1 Evaluation Algorithm 1

Section 3.3.2.1 gives the ERT computation for Evaluation Algorithm 1 as:
$ERT_{1,i,j} = B_{i,j} * P_{Attenuation}(Range(i,j))$

Accessing the value $B_{i,j}$, finding the distance from i to j ($Range(i,j)$)), and computing the attenuation due to range ($P_{Attenuation}(Range(i,j))$) and finding the product of the two can each be done in $O(1)$ time. Thus, the ERT computation for this algorithm is $O(1) + O(1) + O(1) = O(1)$. This computation will require $O(1)$ additional pieces of temporary storage for each intermediary value and the resulting product.

3.6.4.2 Evaluation Algorithm 2 & 3

As discussed in Section 3.3.2.2, the ERT computation for Evaluation Algorithms 2 and 3 is given as:
$ERT_{2/3,i,j} = B_{i,j} * P_{observation} * P_{Attenuation}(Range(i,j))$

As mentioned in Section 3.6.4.1, the product of $B_{i,j}$ and $P_{Attenuation}(Range(i,j))$ can be computed in $O(1)$ time with $O(1)$ temporary storage. The computation of $P_{observation}$ begins with determining the deepest depth (D_{max}) that can be seen, without bathymetric obstruction, in target cell (i,j). As previously described in Section 3.6.3, the computation of D_{max} can be done in $O(D_{range})$ time with $O(D_{range})$ temporary storage. Finally, $(P_{observation})$ is computed as the definite integral over the shape function in $O(1)$ time. Thus, the ERT computation for this algorithm is $O(1) + O(D_{range}) + O(1) = O(D_{range})$. The algorithm will need to use $O(1)$ temporary storage for temporary storage of each of $B_{i,j}$, $P_{Attenuation}(Range(i,j))$, and $P_{observation}$. An additional $O(D_{range})$ temporary storage space is required to compute Line of Sight, and $O(1)$ temporary space to store the resulting integral. Thus, the total temporary storage required is given by $O(D_{range} + 1 + 1 + 1 + 1) = O(D_{range})$.

3.6.5 Optimal Receiver Placement

The selection of the top R_{opt} receiver locations requires the Goodness Grid to have been populated. As previously stated, the selection process is iterative (requiring that suppression (if selected) be applied after each selection), as suppression causes values in the Goodness Grid to be altered. The process of selecting a single receiver location is $O(n^2)$ since the algorithm must consider approximately n^2 possible receiver locations at each iteration. After each location selection, suppression must be applied to the chosen location. Due to the variability of suppression algorithm complexity, a variable $(E_{suppression})$ is used to represent the expected runtime of the suppression algorithm. Thus the computation time required for each selection-suppression step is $O(n^2 + E_{suppression})$. The number of iterations required to run is given by the number of optimal (R_{opt}) and projected (R_{proj}) receiver placements requested. In total, the runtime of the Optimal Receiver Placement step is $[R_{opt} + R_{proj}] * [O(n^2 + E_{suppression})] = O([R_{opt} + R_{proj}] * [n^2 + E_{suppression}])$.

3.6.6 Suppression

As discussed in Section 3.4, the suppression mechanic is applied after the placement of a receiver. The suppression mechanic has several options, each affecting the time and space complexity of the mechanism. To better capture the complexity of the algorithms involved, the following variables are defined:

D_{range}	The Detection Range of a receiver.
$d_{detection}$	The edge size of the Detection Area, equal to $2 * D_{range} + 1$ cells.
f_{supp}	The Suppression Range Factor. A non-negative real number.
r_{supp}	The Suppression Range. Equal to $f_{supp} * D_{range}$
d_{supp}	The square root of the number of cells in the Suppression Area. The edge dimension of the Suppression Area. Equal to $2 * r_{supp} + 1$ cells.

3.6.6.1 Static Suppression Algorithm

As stated in Section 3.4.2.1, the Static Suppression Algorithm multiplies all cells within the Suppression Area by a scalar constant. This is a scalar multiplication that can be done in place, requiring only $O(1)$ extra space, but a multiplication operation for each cell in the Suppression Area, which is $O(d_{supp}^2) = O([2*f_{supp}*D_{range}+1]^2) = O([4*f_{supp}^2*D_{range}^2+4*f_{supp}*D_{range}+1]) = O(f_{supp}^2*D_{range}^2)$. In most cases (excluding those where very sparse acoustic networks are desired), the Suppression Range Factor, f_{supp}, will be rather small (less than 2.0), which reduces the runtime complexity to $O(D_{range}^2)$.

3.6.6.2 Exact Suppression

As stated in Section 3.4.2.3, the Exact Suppression Algorithm uses the user-specified Evaluation Algorithm to calculate the number of transmissions that have already been observed and should be discounted from future consideration. Here, E_x denotes the expected runtime for Evaluation Algorithm x's computation of a single cell's ERT and, T_x denotes the expected temporary storage requirement for Evaluation Algorithm x's single-cell ERT computation. The Exact Suppression Algorithm is broken into three distinct steps: Suppression Area ERT computation, Behavior Grid Update, and Goodness Grid Recalculation.

Suppression Area ERT Computation
The Suppression Algorithm first determines the ERT for each cell within the Suppression Area, and stores them in the ERT Area, a temporary Grid with the same dimension as the Suppression Area. This step is similar to the Evaluation process, with the exception that the Suppression Area's dimensions differ from those of the Detection Area by a factor of f_{supp}. The computation of the ERT Area requires E_x time and T_x temporary space for each of the d_{supp}^2 cells in the Suppression Area. An additional $O(1)$ temporary space will be needed to store the resulting ERT values for each cell. Therefore, the ERT computation will require $O(E_x * d_{supp}^2)$ time and $O((1+T_x) * d_{supp}^2) = O(d_{supp}^2 * T_x)$ temporary storage.

Behavior Grid Update
Next, the suppression algorithm subtracts the ERT Area from the corresponding area in the Behavior Grid. This is a simple subtraction operation between each of the d_{supp}^2 pairs of corresponding cells in the Grids. Because each subtraction takes $O(1)$ time and temporary storage, the time and space complexities are both given by $O(1) * O(d_{supp}^2) = O(d_{supp}^2)$.

Goodness Grid Recalculation
Finally, the Goodness values of all cells within Detection Range of the Suppression Area must have their Goodness value updated (as the ERT of one or more cells within their Detection Area has just been reduced). The area affected by suppression, and therefore requiring recalculation, is a square with edge diameter $2 * (d_{supp} + D_{range}) + 1$. Each cell in this area will require $O(d_{detection}^2 * E_x)$ time to re-compute its Goodness Value. The total time for updating the Goodness Grid is then: $O([2 * (d_{supp} + D_{range}) + 1]^2 * O(d_{detection}^2 * E_x)) = O((d_{supp} + D_{range})^2 * d_{detection}^2 * E_x)$. The

Goodness Grid update will require $d_{detection}^2 * (T_x + 1) = O(d_{detection}^2 * T_x)$ temporary storage to compute and store intermediate ERT values. This temporary storage can be recycled for each sequential Goodness Cell computation.

The total computation time for the Exact Suppression Algorithm is then given by O(ERT Computation) + O(Behavior Grid Update) + O(Goodness Grid Recalculation). This is a total of $O(E_x * d_{supp}^2) + O(d_{supp}^2) + O((d_{supp} + D_{range})^2 + d_{detection}^2 * E_x)$ time. The algorithm will also require $O(MAX(d_{supp}^2, d_{detection}^2 * T_x)$ temporary storage, which can be recycled between each of the above stages.

CHAPTER 4 IMPLEMENTATION

This code base is intended to be treated as a tool box for the optimization of acoustic networks. While a fully-functional application is provided, its intent is to serve as a template for integrating various functionalities. It is expected that researchers will want to define and use customized versions of the provided functions (herein referred to as “subfunctions”). To support this, the framework utilizes generic “dispatcher functions” and dictionary-based parameter passing.

4.0.1 Parameter Dictionary

The framework takes as an input to the main program a dictionary of named values (`params`), and passes it throughout the program. This allows for dispatcher functions with concrete function signatures (volatile parameters can be incorporated into the parameter dictionary), and sub-functions with widely varied signatures. The benefit of this is that the function signatures of the dispatcher functions will rarely need to be changed, leading to a stable but flexible framework. The cost of this approach is that data validation of dictionary-based parameters must be preformed before the execution of sub-functions in order to avoid run-time errors within the varied sub-functions.

4.0.2 Sub-Functions

Section 3 discusses the framework’s conceptual models such as Goodness computation and animal modeling. These models have loose operational requirements that can be thought of as roles. For example, it is expected that an animal model populates the Behavior Grid, and that an Evaluation Algorithm populate the Goodness Grid. As long as a user-customized sub-function fulfills its role within the prescribed model (suppressing a given location, populating a Grid, etc.), it can be easily integrated into the framework.

4.0.3 Dispatcher Functions

The primary purpose of dispatcher functions is to create a uniform interface within the framework for calling various sub-functions. Each model has a corresponding dispatcher function, and these dispatcher functions are responsible for preparing and calling sub-functions which belong to that model. For example, the Animal Model dispatcher is responsible for functions which populate the Behavior Grid. A dispatcher function performs a series of if-else checks over one or more “option” variables in the parameter dictionary, and executes the corresponding sub-function. Once a user defines a new sub-function, they should add the new sub-function as an option within the corresponding dispatcher. This will allow a user to use the new sub-function by calling the dispatcher function with the new option. Any sub-function-specific data validation should be done at the beginning of the program, before any computationally-expensive operations are preformed, by the `checkParams()` function (Section 4.0.4.1).

4.0.4 Major Modules

4.0.4.1 Parameter Checking

As previously mentioned in Section 4.0.1, the cost of flexible inputs is substantial data validation. Within the framework, data validation should occur early on so that errors can be identified before too much time has been invested in computation or data-loading. To this end, the `checkParams()` function is provided to handle all data validation and reporting. The function takes in the `params` parameter dictionary and a `stop` boolean (set to True by default). The function and validates all internal dependencies, reporting any errors found, and halting the program if `stop` is True. The function also assigns necessary default values if no user-defined values are provided. The function returns the validated `params` parameter dictionary (with default values if necessary).

4.0.4.2 File Output

The File Output Module is responsible for generating graphical and textual results for the framework. The primary function in this module is the `graph()` function, which takes in three parameters, a `result` parameter dictionary, a `params` parameter dictionary, and a boolean, `showPlots`. The `result` parameter dictionary, contains results (Table 4.1) from successfully completing a simulation. The `params` parameter dictionary contains parameters passed into the program, and generated by the framework. `showPlots` directs file output. If `showPlots` is set to False, the function writes the output files described in Section 5.1 to disk, and returns a dictionary of file paths that point to the the newly written files. If `showPlots` is set to False, then the same results are displayed within an R session instead of being written to disk. Filenames for output files follow a `timestamp-FileName` convention, where `timestamp` is a user-specified string (defaulting to the time the request passed Parameter Validation) and `FileName` is a contextual identifier ("GoodnessGrid", "BehaviorGrid", etc). Graphic output from this module is generated using the R `graphics` package. Zip file creation uses the R `zip` package.

4.0.4.3 Bathymetry Parsing

The Bathymetric Parsing Module is responsible for the parsing, loading, and validation of bathymetric data, with the ultimate purpose of generating a Bathymetry Grid. The dispatcher function for this module is the `getBathy()` function, which requires a `params` parameter dictionary containing the key-value pairs described in Table 4.2. The function returns a validated Bathymetry Grid of `XDist` columns and `YDist` rows. As previously mentioned in Section 3.2.2.2, there are a multitude of data formats for bathymetric data, each requiring a unique data parser. To support

Key	Value	Description
`topographyGrid`	The Bathymetry Grid	An alias for the unmodified Bathymetry Grid.
`behaviorGrid`	The Behavior Grid	The unmodified Behavior Grid.
`goodnessGrid`	The Goodness Grid	The unmodified Goodness Grid.
`coverageGrid`	The Coverage Grid	The coverage computed from the resulting receiver array.
`recoveryRates`	Receiver array data	A dictionary of receiver array-related results obtained as an output from the simulation. The dictionary is generated by `sensorFun()`.
`stats`	Receiver array statistics	A dictionary of receiver array-related statistics obtained from calling getStats().

Table 4.1: A summary of the key-value pairs in the `result` dictionary.

s

this diversity of formats, the `getBathy()` function reads the `inputFileType` from the `params` dictionary to select an appropriate parser. The selected parser begins reading data at column `startX` and row `startY` from the data file specified by `inputFile`. The `getBathy()` function also checks to makes sure that all cells in the Bathymetry Grid have rational depth values. N/A, INF, or -INF values are invalid as they will contaminate future computations, resulting in runtime errors. The module replaces invalid values with a value of '0'. By default, the framework supports the NetCDF type, and thus requires the `ncdf` or `ncdf4` R packages. Both the `ncdf` and `ncdf4` packages serve as an R interface for the NetCDF C/FORTRAN library, and thus requires that a working NetCDF distribution be installed on the system.

Key	Value	Description
`inputFile`	File path to Bathymetry File.	A relative or absolute Bathymetry File path.
`inputFileType`	Parser name.	Indicates which parser to use to read the Bathymetry File.
`XDist`	Bathymetry Grid column count.	Number of columns in the Behavior Grid.
`YDist`	Bathymetry Grid row count.	Number of rows in the Behavior Grid.
`startX`	Column index to read from.	The column index in the Bathymetry File to begin reading from.
`startY`	Row index to read from.	The row index in the Bathymetry File to begin reading from.

Table 4.2: A summary of the `params` key-value pairs used in Bathymetry Parsing module.

4.0.4.4 Animal Modeling

The Animal Modeling module is responsible for simulating animal distribution, and ultimately generating the Behavior Grid. The dispatcher function for this module is the `fish` function, which takes in a Bathymetry Grid and a `params` parameter dictionary containing the key-value pairs described in Table 4.2.

As previously mentioned in Section 3.2.3, the Animal Model offers separate horizontal and vertical movement models. The animal movement models are handled by the `fish()` function, which requires a `params` parameter dictionary. The `fishModel` dictionary key specifies which horizontal movement model should be employed by the module to generate a population distribution. Because the vertical movement models operate independently of the horizontal movement model and each other, separate option variables for both vertical movement models are provided. The Restricted Vertical Habitat Range model is used if two keys, (`mindepth` and `maxdepth`), are specified within the `params` dictionary. Similarly, the Habitat Preference model is used if `depth_off_bottom` and `depth_off_bottom_sd` keys are present within the `params` dictionary.

Key	Value	Description
`fishModel`	Behavior Model Name.	The chosen Horizontal Behavioral Model.
`mindepth`*	Min depth for animal habitat.	Minimum depth in the Restricted Vertical Habitat Range Model.
`maxdepth`*	Max depth for animal habitat.	Maximum depth in the Restricted Vertical Habitat Range Model.
`depth_off_bottom`*	Preferred animal habitat depth.	Preferred depth in the Habitat Preference Model.
`depth_off_bottom_sd`*	SD of preferred depth.	Standard Deviation of preferred depth in the Habitat Preference Model.

Table 4.3: A summary of the `params` key-value pairs used in Animal Modeling module.
* optional parameters.

4.0.4.5 Goodness Computation

The Goodness Computation module is responsible for computing the goodness of a particular cell (Section 3.3). The dispatcher function for this model is the `goodnessGridFun()`, which takes in `params`, a parameter dictionary, and `grids`, a dictionary containing the Bathymetry Grid, and Behavior Grid. The `bias` option is used to tell the module which sub-function should be used to compute Goodness. All necessary parameters for the chosen sub-function are passed in via the `params` dictionary. The module is responsible for computing and adding the GoodnessGrid to the *grids* dictionary. Unlike the Animal Population and Bathymetric parsing modules, this module does not directly return a Grid object, but inserts it into the `grids` dictionary. This behavior reflects the idea that the Goodness is logically dependent upon bathymetry and animal behavior. Furthermore, this behavior is intended to guide users towards creating a Bathymetry and Behavior Grid before generating a GoodnessGrid.

Because the Goodness computation can take a substantial amount of time, sub-functions which compute Goodness should inform users what percentage of the computation has been completed. The `silent` parameter is a boolean that disables this update when set to True.

4.0.4.6 Suppression

The Suppression module is responsible for suppressing (Section 3.4) the Goodness of the area around a particular cell on the Goodness Grid. The dispatcher function for this module is the `suppressionFun` function, which takes in `params`, a parameter dictionary, `grids`, a dictionary containing the Goodness, Bathymetry, and Behavior Grids, and `loc`, a dictionary mapping the keys 'r' and 'c' to the row and column indexes of the cell to be suppressed. The module is responsible for reducing the values around an area, given by `loc`, on the GoodnessGrid and returning the suppressed GoodnessGrid. Note that the suppression module is only responsible for performing suppression operation on a single target location, `loc` (recall that the process of selecting a receiver location and suppressing it is an iterative process, and therefore must be interleaved). The method in which suppression is applied is given by the `suppressionFcn` option in `params`. The dimensions of the square area affected by suppression is given by `suppressionRange`. Section 3.4.2 describes how `minsuppressionValue`, `maxsuppressionValue`, and `shapeFcn` can be used to define a suppression template or gradient.

Key	Value	Description
`suppressionFcn`	Suppression Function Name.	The name of the suppression
`suppressionRange`	Distance from loc to suppress.	How far (in cells) out from loc to suppress.
`receiverElevation`	Elevation off sea floor.	Sensor elevation (in meters) off the see floor.
`minsuppressionValue`*	Min suppression multiplier.	Minimum penalizing coefficient.
`maxsuppressionValue`*	Max suppression multiplier.	maximum penalizing coefficient.
`shapeFcn`*	Distribution Function ID.	Name of the statistical distribution model to use.

Table 4.4: A summary of the `params` key-value pairs used in the Suppression module.
* optional parameters.

4.0.4.7 Sensor Placement

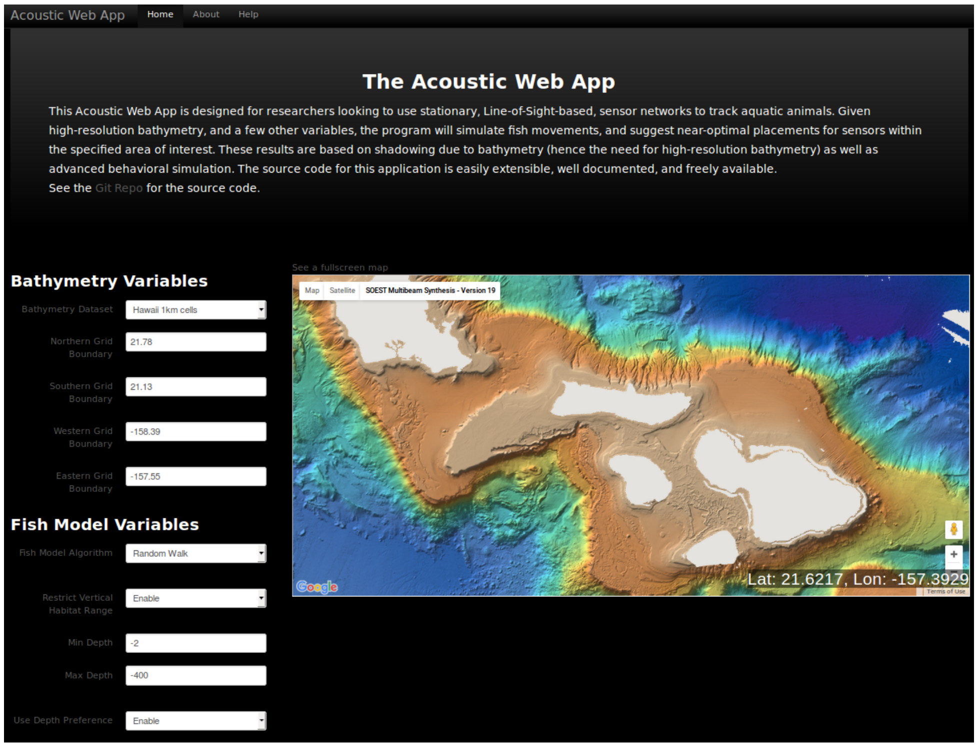
As mentioned in Section5.3.2, a tight coupling exists between the heuristic and Evaluation Algorithms. The sample application provided is very basic, utilizing a brute-force goodness computation on all cells in a the study area, and an exhaustive search to identify potential candidates for receiver placement. Also recall that the selection of receiver placements and suppression are an iterative process. To satisfy both of these relationships, the three operations are combined into a larger Sensor Placement module. The dispatcher function for this module is `sensorFun`, which handles goodness computation, the selection of receiver locations, and the suppression of those locations. It is important to note that while this function is important and warrants mention, it is not an extensible "module". That is to say, it is merely a driver function that binds together three independent modules (Goodness Computation, Sensor Placement and Suppression), handling the (minimal) dependencies between each. Modifying the nature in which receiver placements are chosen for consideration will require a customized driver. The `sensorFun` function takes in `grids`, a dictionary containing the Bathymetry and Behavior Grids and a parameter dictionary, `params`. The module is responsible for outputting the `result` dictionary, which contains the Bathymetry, Behavior, and Goodness Grids, a list of receiver placements (both user and system placed receivers), statistics regarding those receivers (absolute and unique data recovery rates, and a delta value for the array), the `params` parameter dictionary, and a list of warnings/errors.

4.0.4.8 Sample Application

To provide a demonstrable product, and program outline, a simple, cohesive application, `acousticRun`, is provided. The application makes use of framework functions to validate parameters, construct Bathymetry, define Behavior, compute Goodness, and choose optimal receiver locations with suppression. The application supports all initial functionality provided in the framework, and allows users to call these functions over their own datasets. As previously noted, the application implements a brute-force computation of goodness for every cell in the Goodness Grid, and employs an exhaustive search to select optimal receiver locations. Due to this, the application is suitable for small, sample simulations, but might not be appropriate for very large scale simulations. Documentation [5] for the application lists all available parameters, descriptions, and constraints. Due to the large number of parameters available, the program also provides a "default" simulation when no parameter are given. The `acousticRun` function provides an entry point for users into the framework. Typically, this would involve a user loading the `acoustic` package in R, reading the documentation (a wiki page about parameters), providing some or all of the parameters described, and calling `acousticRun` with those parameters (as a dictionary). Once the simulation finishes, users will be presented with written and graphic results.

4.0.4.9 Web Server

Included alongside the framework is a web-application (webapp) with a Graphical User Interface (GUI) that runs a simple demonstrative (demo) application using the framework. The `Rook` R package is used to handle HTTP interactions between the GUI and the R demo application. The GUI is a simple HTML page that generates a list of parameters (including a unique *timestamp* identifier) based on user input, creates a JSON (JavaScript Object Notation) string, and sends an HTTP request to the Rook server with that JSON string. Using the `rjson` library, the Rook server decodes the JSON string into R data objects, and calls the demo application using the decoded


Acoustic Web App
Home
About
Help
See a fullscreen map
Map
Satellite
SOEST Multibeam Synthesis - Version 19
Lat: 21.6217, Lon: -157.3929

The Acoustic Web App

This Acoustic Web App is designed for researchers looking to use stationary, Line-of-Sight-based, sensor networks to track aquatic animals. Given high-resolution bathymetry, and a few other variables, the program will simulate fish movements, and suggest near-optimal placements for sensors within the specified area of interest. These results are based on shadowing due to bathymetry (hence the need for high-resolution bathymetry) as well as advanced behavioral simulation. The source code for this application is easily extensible, well documented, and freely available.
See the Git Repo for the source code.

Bathymetry Variables

Bathymetry Dataset: Hawaii 1km cells
Northern Grid Boundary: 21.78
Southern Grid Boundary: 21.13
Western Grid Boundary: -158.39
Eastern Grid Boundary: -157.55

Fish Model Variables

Fish Model Algorithm: Random Walk
Restrict Vertical Habitat Range: Enable
Min Depth: -2
Max Depth: -400
Use Depth Preference: Enable

Figure 4.1: A screen shot of the webapp GUI. On the right is a Google Map widget that assists users in choosing latitude and longitude boundaries for their study site (defining their Bathymetry Grid). On the left is a list of simulation options implemented as drop-down lists, and simulation parameters implemented as text boxes. At the bottom (hidden) is a "Submit" button that sends that validates the parameters and sends them to the server. All parameters fields and lists display "tool tip" style descriptions of the parameter that field/list represents.

parameters. The Rook application signals the successful reception and decoding of the JSON string as an HTTP 200 status update. The GUI then begins waiting for the completion of the simulation (which may take some time) by continuously checking (ignoring failures) for the existence of a text document (named for the unique *timestamp* identifier) on the server. When the simulation finishes its execution (either via error or normal completion), it writes its several output files (Section 5.1), and a text document containing the JSON representation of the outputs (file paths to graphics and a dictionary of results). All output files are tagged with a unique *timestamp* identifier. Once the GUI finds an output file with the correct *timestamp* signature, it reads the file, parses the results, and displays them within an HTML page. If the local system supports it, a zip file of all result files is also supplied. The webapp serves to simplify the selection and specification of parameters to the demo application. This allows users to quickly and easily understand the utility of the framework and application.

4.0.5 Implementation Hurdles

4.0.5.1 Complexity and Usability

The primary trade-off that occurs with any application is that between simplicity and control. While a simpler, less-detailed application may lower the effort required, it inevitably reduces the level of control a users has over the application. A significant number of variables and options may serve to increase control over an application, but also require increased time and effort to understand and use effectively. Put simply, very specific actions require very detailed instruction, which in turn requires significant effort. On the other hand, simpler applications may be more easily understood and adopted, but ultimately lack higher levels of control. To address this trade-off, both a demo application (Section 4.0.4.9) and framework are provided. The demo application provides an easy-to-use application that works "out-of-the-box" and has a useful subset of features, while the framework provides access to more detailed functionalities for advanced users.

4.0.5.2 Coordinate Conversion

As mentioned in Section 3.2.2.1, bathymetric files are composed of a regular grid of cells. However, because the Earth is a 3D sphere, translation is necessary to map that 3D sphere to a 2D grid. This translation introduces potential error into the specification of a study site. One such source of error occurs when the method of translation is not specified, leading to ambiguity in how grid cells should be read. Another possible source of error can be attributed to rounding errors in the number of degrees latitude and longitude represented by each cell. In grids spanning hundreds of thousands of cells, small rounding errors can lead to large positional discrepancies. To deal with these issues, the framework requires users to specify their study site as grid cell indexes. This puts the burden of translating latitude and longitude into gird coordinates on the user (who likely knows best how to handle their data). Additionally, final receiver placements are given in terms of global (relative to the input bathymetric file) and local (relative to the defined study site) indexes. This facilitates error checking and helps to identify translational errors.

4.0.5.3 Rook

The *Rook* package was selected as a web server because it offered support for cloud-based web hosting on Heroku and Amazon Web Services (AWS). This feature was desirable as it allowed users to quickly create a working instance of the web application without modification to their local computers. Unfortunately, the *Rook* web server is implemented using a single, blocking thread. This means that *Rook* is only able to handle one request at a time, and will wait for a request to complete before accepting any other other requests. Because the same Rook thread that receives a request must also call the demo application (which takes about 30 seconds to complete), the entire web server is unresponsive to new requests until the demo application completes its execution.

This unresponsiveness can lead users to feel like the application is “frozen” or “broken”. To solve this problem, a loading icon (spinning circle) is used to assure users that the webapp is working. Another issue with the *Rook* package is that it imposes a 250 character limit on the length of POST request data. Because of the substantial number of parameters sent to the demo application, POST data strings are often very long ($\geq$500 characters). To compensate, POST data (the JSON string representing user-specified parameters) from the HTML page must to be cut into shorter segments and sent with sequential identifiers and re-assembled on the server.

CHAPTER 5
RESULTS

5.1 Output Files

The program outputs 6 files:
1) The Bathymetry Grid as a heat map
2) The Behavior Grid as a heat map
3) The Goodness Grid as a heat map
4) The Coverage Grid as a heat map
5) The marginal gain in Unique Recovery Rate as a function of the number of receivers placed
6) Text representations of the 5 Grid files, tabular receiver data, and the specified input parameters.

5.1.1 Grid Graphs

The four Grid files (Bathymetry, Behavior, Coverage, and Goodness) are visualized as a heat maps. All of these are overlaid with the resulting receiver locations as numbered circles. Receiver locations show their detection range as dotted circles centered over the receiver locations. User placed receivers are colored gray, receivers placed optimally by the system are colored blue, and projected receivers (also placed by the system) are colored green. The numbering on receivers denotes the rank of each receiver's Unique Data Recovery Rate. User and system-placed receivers are ranked separately. The highest ranked system-placed receiver locations are returned as the optimal receiver locations, with the lower ranked locations as the projected receiver locations.

5.1.2 Data Recovery Graphs

The program also produces a graph of the marginal increase in and cumulative sum of the Unique Data Recovery Rate as a function of the number of optimally placed receivers used (Figure 5.1). The graph of the marginal increase in UDRR (the lower graph) is especially useful. Given that Receivers have a fixed per-unit cost, it makes sense to weigh a receiver's effectiveness (in this case, UDRR contribution) against its cost to determine the utility of purchasing an additional receiver. The graph of cumulative UDRR is useful to quickly identify the number of receivers necessary to reach a goal UDRR.

5.1.3 Text Files

The program returns a comprehensive text representation of the program output (text dump of gridded data and input parameters), and a short, human-readable document that lists the primary

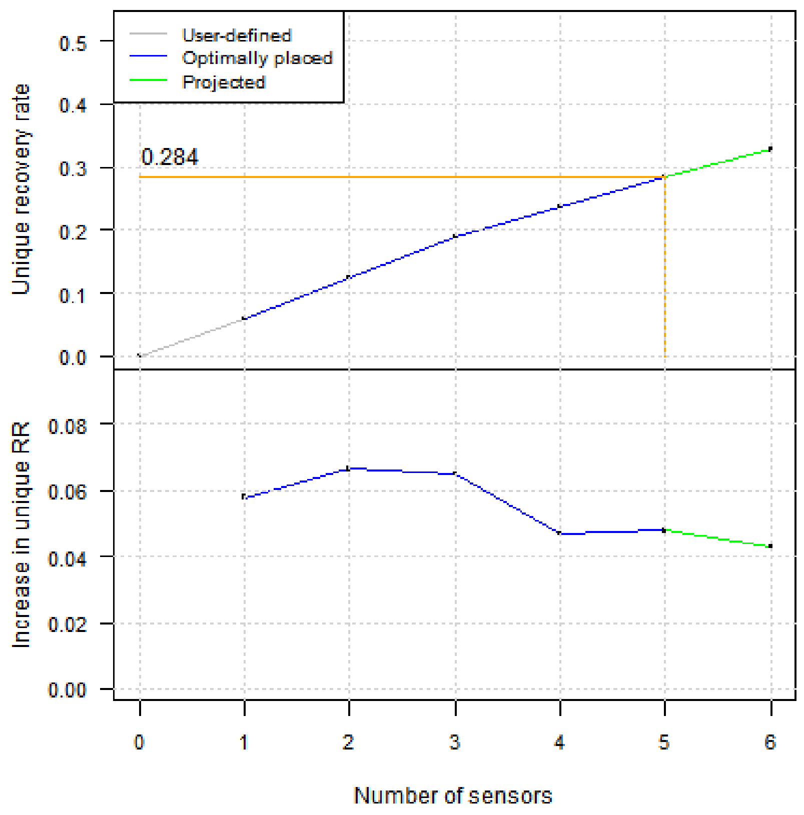


Figure 5.1: A graphical representation of the cumulative and per-receiver UDRR. User-placed receivers are representd by a grey line, system-placed receivers are represented by the black line, and projected receivers are represented by the green line.

simulation parameters (Evaluation Algorithm, Suppression Algorithm, Behavioral Model, Input Grid Size, and Detection Range), as well as a tabular output of receiver placements (coordinates), data recovery rates for each receiver, and network sparsity. Projected receivers are excluded from the total UDRR, ADRR, and sparsity values.

Coordinates are returned in both a Global (with respect to the original Topography file) and local (the user specified area of interest) frame. While the curvature of the earth is well documented, different bathymetric maps may handle the mapping of a 3D curved plane to a 2D Grid differently. For example, one Grid may implement some scaling on Grid cells a function of the cell's latitude or distance from a certain point. Other Grid files may simply provide non-square Grid files. By proving a small-scale (local) and large-scale (global) frame of reference for output receiver locations, irregularities of this nature are more easily detected.

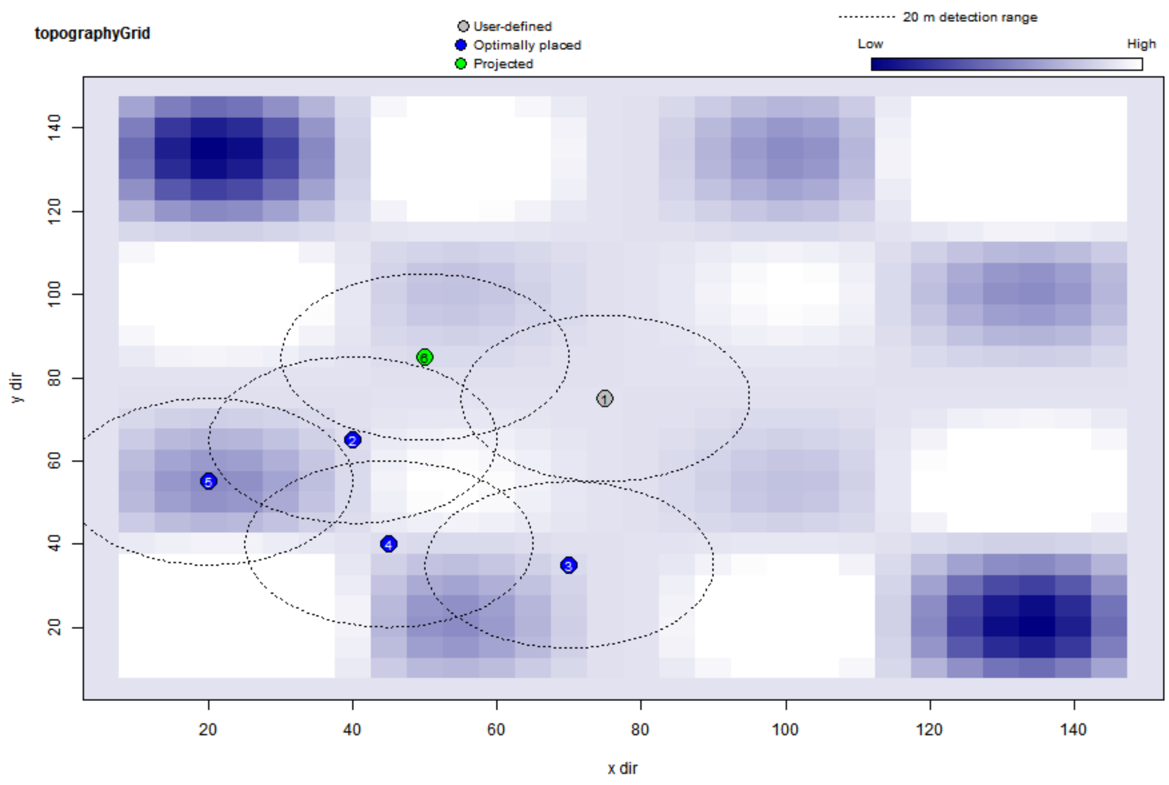


Figure 5.2: A graphical representation of an artificial Bathymetry file at a 1m resolution (each cell has an edge length of 1m). Darker cells represent greater depths, while white cells represent inaccessible terrain (dry land). The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed receivers as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines.

5.2 Distinguishing Characteristics

The challenge of acoustic network design is one that is common within the acoustic tracking community, yet rarely addressed. While there are many methodologies, applications, and services that focus on achieving various objectives, this framework distinguishes itself in its availability, and adaptability.

5.2.1 Availability

As previously stated, MANDe is freely available [4] and falls under the GNU General Public License (GPL 2). This availability means its methodologies are open-sourced, fully transparent, and are available for peer-review. Furthermore, MANDe is accessible to researchers without requiring expertise in R. The web application provides a useful set of well-documented functionalities with a user-friendly interface.

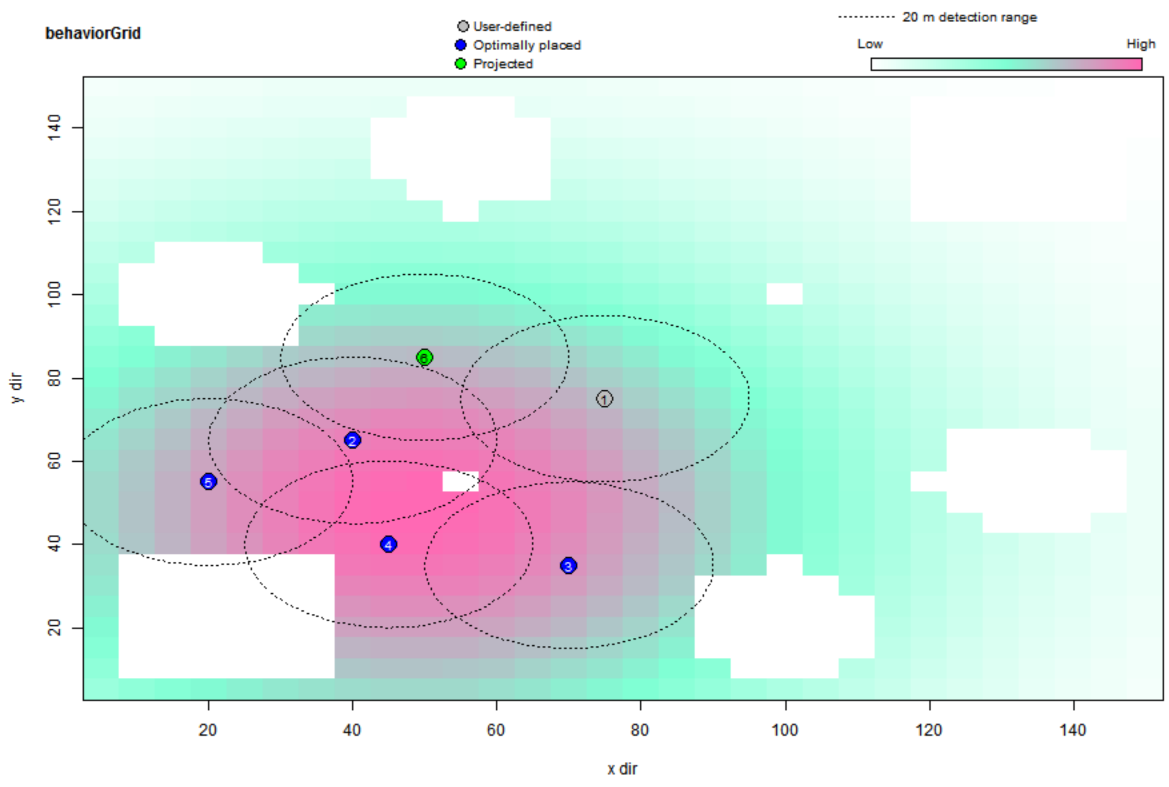


Figure 5.3: The Behavior Grid represented as a heat map Higher levels of animal residency correspond with pink cells, moderate levels as light blue, and white for non-residency (inhospitable habitat such as dry land). The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed receivers as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines.

5.2.2 Adaptability

5.2.2.1 Flexability

Many of the modules in this framework can be used as stand-alone functions that perform useful operations (reading bathymetric data, calculating distribution densities, etc). This modularity allows the framework to function more like a library for acoustic simulation than a fixed application.

5.2.2.2 Simple Data Formats

Many of MANDe's key functionalities (Evaluation Algorithms, Suppression, Animal Modeling, Network Statistics) operate over very simple data containers (Grids). This simplified data format makes it easy to import, export, and convert between other data formats. This simplified format also makes it easy to move data between MANDe and other R scripts.

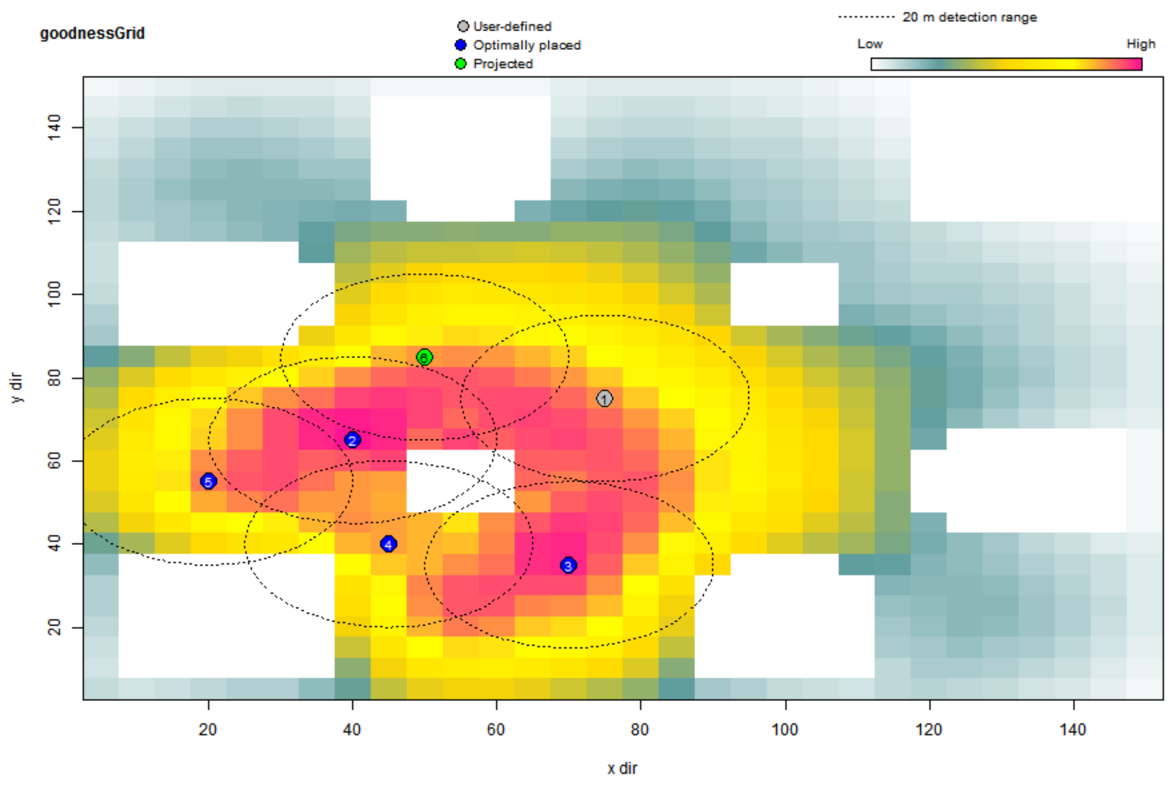


Figure 5.4: The Goodness Grid represented as a heat map showing the sum total of Estimated Receivable Transmissions (ERT) for a receiver placed in a particular cell. The legend in the top right assigns color coding to various ERT values. The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed receivers as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines.

5.2.2.3 Customizable

While other applications and methodologies optimize for fixed-objective functions (k-neighbor coverage, fixed detection probability, etc), MANDe allows for user-customized objective ("Goodness") functions. Section 4.0.3 discusses dispatcher functions, which serve as abstract entry points into stand-alone modules, and allow highly-customized sub-functions to be aggregated together into uniform modules. These dispatcher functions makes it easy to integrate new functionality into the framework without affecting other sub-functions.

5.2.3 Drawbacks

5.2.3.1 Explicit Data Validation

In most all software systems, future efforts to add new functionality require a different set of parameters than those initially specified. Thus, future developers will likely need to modify function

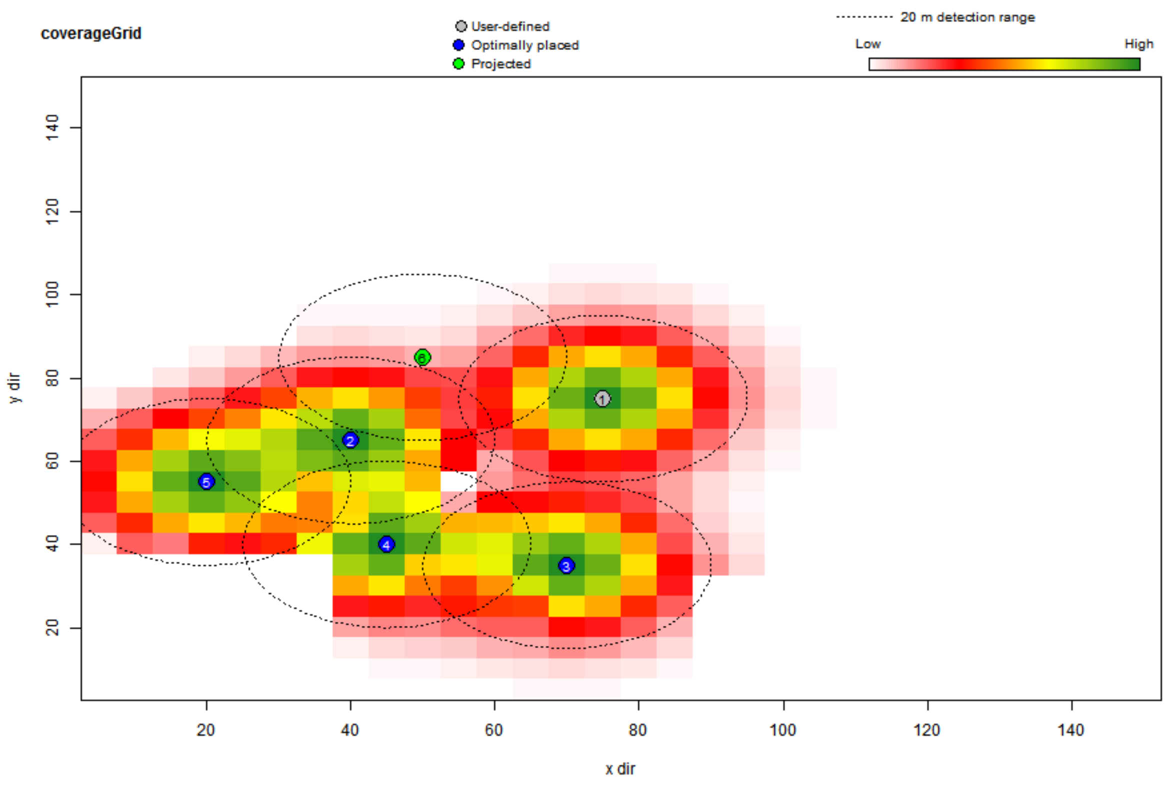


Figure 5.5: The Coverage Grid represented as a heatmap showing the quantity of Estimated Receivable Transmissions from each cell in the study site, for the designated receiver array configuration. The legend in the top right assigns color coding to various ERT values. The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed receivers as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines. The missing corner of of Receiver 4's Detection Area is due to the presence of an obscuring section of Bathymetry (dry land).

signatures to incorporate said functionality. This modification can substantially impact the way users interact with the system, and will require other developers to become familiar with a modified interface. Additionally, these changes can potentially break older functions that relied upon the old function definition. To reduce this brittleness (a small change causing a system to break) associated with explicit parameters, dictionary-based parameter passing is employed. While explicit parameters cause brittleness within a system, they serve to create a contract between users and functions, such that if users provide those explicit parameters, the function will operate as intended. This contract is enforced by the programming language, resulting in compilation/interpretation errors when these parameters are violated. Without explicit parameters, this contract is gone, and the responsibility of enforcing the relationship (by way of key-value validation within the dictionary) between users and functions falls to either the user or the developer. In short, dictionary-based parameter passing provides reduced brittleness and simplified future-development but passes the responsibility of data validation to users and developers.

5.2.3.2 Technical Capability

While a fully functional application is provided as a demonstration of this framework, the framework's greatest usefulness to a tracking project will likely not be realized unless custom Goodness functions are defined. Because substantial expertise in programming is required to author these custom functions, the framework is less useful to users with little R experience.

5.2.3.3 Language Limitations

This framework was implemented in the R programming language primarily due to its popularity within the biological community. While R is sufficient for small and medium scale simulations, a compiled language would offer significantly faster execution at all simulation levels. Additionally, the R platform does not offer good support for reverse compatibility. Specifically, R packages need to be recompiled under the local system's version of R. If this compilation is unsuccessful (perhaps due to a difference in R versions, or operating system limitations), the package will not function. If an application requires a significant number of R packages, then it is likely to be limited to operating within a specific version of R.

5.3 Future Work

5.3.1 Empirical Analysis

The analytical framework described herein is largely theoretical, despite a basis in well documented physical phenomena. In order to better understand the correctness of this framework, it would be useful to analyze empirical data from acoustic networks, of varying scales, designed with it.

5.3.1.1 Acoustic Coverage

Recall that estimated data recovery rates are calculated based upon acoustic coverage and animal distribution. While basic Animal Models are provided, the accuracy of these models (and as a result the accuracy of the resulting data recovery rates) is very species-dependent and outside the scope of this paper. While data recovery rates are based on animal models, the acoustic coverage model is not. Thus, the accuracy and optimality of MANDe's bathymetric shadowing and network coverage models is a priority. It would be interesting to design a real-world acoustic network with MANDe, and then compare the actual and theoretical coverage.

5.3.1.2 Workflows

The workflows described in Section 3.1.2 were based upon both personal experience and imagined use. It would be useful to investigate the actual use cases, workflows, and problems that researchers face at various points throughout the lifetime of an acoustic tracking project. By identifying unconsidered problems and workflows, the framework can be modified/expanded to encompass new roles and features, leading to greater usefulness.

5.3.2 Heuristic Algorithms

Recall that the function of the Goodness Grid is to store Goodness values for various locations within the study area. The sample application simply computes every possible goodness value in the Goodness Grid, and then finds the top values Goodness values via R's $max()$ function. While this process is guaranteed to find optimal receiver positions, it is computationally intensive, accounting for the vast majority of the program's run time. Utilizing heuristic algorithms can significantly reduce the number of Goodness values that are computed, but can also reduce the optimality of the final acoustic array. Heuristics are currently not included in the framework because functions involving Evaluation Algorithms and any new heuristic algorithms will likely be tightly coupled (a heuristic algorithm will likely need to consider an Evaluation Algorithm's definition of *Goodness*). Furthermore, to avoid computational overhead (such as passing large lists of heuristic candidate cells), it is likely that the heuristic algorithm and Evaluation Algorithm will be called from within the same function. This will likely lead to a large number of customized functions that implement heuristic algorithm-Evaluation Algorithms combinations. It would be very useful to evaluate the cost (as program execution time) of programatically de-coupling Evaluation and heuristic algorithms and passing large lists of heuristic candidates. This data could then be compared against the measured benefits of various heuristic algorithms to evaluate the best methodology for integrating heuristics into the framework. Another useful avenue would be the investigation of Evaluation Algorithm agnostic heuristics, which could provide better run times without tight couplings.

5.3.3 Interactive Visualizations

It could be useful to allow users to interact with the output graphs as a graphical user interface. Users would able to move receivers around the graph and see how new placements affects network statistics in real-time. Given that the Goodness Grid can be pre-computed, moving receivers and updating the effects of that movement in real-time requires very little re-computation. In this case, it might be unnecessary to even pre-compute the Goodness Grid, but instead only preform the Evaluation Algorithm on cells where a user places a receiver. This would allow users to quickly see how various receiver arrangements affect their network design, and facilitate rapid, impromptu prototyping of various configurations.

Utilizing 3D visualizations could further enhance the user's understanding of network topology. Figure 5.6 illustrates a first-person view of the 3D environment modeled by the program. Here, bathymetry is rendered as 3D mounds, with animal distributions represented by green fish, and receiver coverage by red spheres. Users might be able to "swim" through the 3D environment, viewing their network from various angles. Such a visualization would help users better understand what kind of animal behavior is being modeled, how their array is laid out, and where modifications might be helpful.

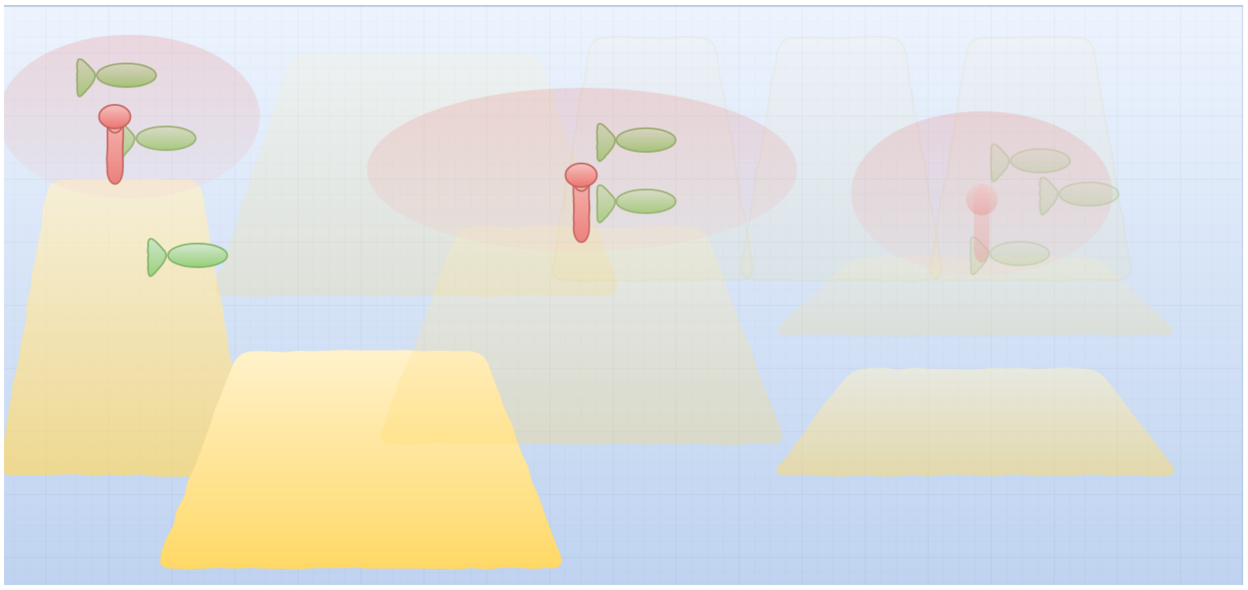


Figure 5.6: A first-person view of the 3D space modeled in the program. Users would be able to "swim" through the environment and gain a better perspective on what they are modeling, what their receiver array might look like in the environment, and where more receivers might be needed.

CHAPTER 6
CONCLUSIONS

MANDe is a framework for the optimization and measurement of Static Acoustic Observation Networks (SAONs), the benefits of which are decreased cost of data collection, increased volume of collected data, and higher data quality. While these benefits are largely realized during the design of a SAON, the framework's utility extends to the augmentation and assessment of acoustic tracking studies. The ability to include an already existing SAON in the design process allows for the optimal augmentation and integration of SAONs. This is useful to projects that operate on annual budgetary allotments and periodically purchase and deploy of hardware. By supporting this incremental growth pattern, MANDe can be used to make optimal use of limited resources. This functionality also provides a mechanism for researchers from separate studies to pool resources, mesh disjoint networks, and collaboratively increase data recovery rates. MANDe also provides automated metrics for the evaluation and comparison of SAONs. These metrics facilitate the evaluation of existing and theoretical network topologies, which is useful when demonstrating the need for additional hardware.

Finally, MANDe is customizable, allowing users to develop their own Behavior, Evaluation, and Suppression Algorithms, Shape/Attenuation Functions, and metrics. Coupled with the fact that the framework is open source (the source code is freely available and editable [4]), and free to use, it is expected that this tool will have a strong impact on the acoustic tracking community.

APPENDIX A
TABLE OF NOTATIONS

A	Bathymetry Grid - A 2D Grid of non-negative real numbers that describe the depth of a section of the sea floor. $A_{i,j}$ refers to the value in the cell at row i, column j. A_x refers to the value in cell x.
B	Behavior Grid - A 2D Grid of non-negative real numbers that indicate the percentage of all transmissions that are expected to be released from that cell over the entire experiment. $B_{i,j}$ refers to the value in the cell at row i, column j. B_x refers to the value in cell x.
G	Goodness Grid - A 2D Grid of real numbers with the same dimensions as the Bathymetry Grid. Contains real numbers that indicate how good a particular cell will serve as a receiver placement. $G_{i,j}$ refers to the value in the cell at row i, column j. G_x refers to the value in cell x.
n	The square root of the total number of cells in the Bathymetry Grid (A). Approximates the edge dimension of a square Bathymetry Grid.
k	The fixed elevation of receivers off of the sea floor in meters. Assumed to be constant for all receiver emplacements in the simulation.
$ERT_{x,T,i,j}$	Estimated Receivable Transmissions - The number of transmissions that are expected to be received from a cell at row i, column j by a receiver at cell T, according to Evaluation Algorithm x.
$P_{Attenuation}(x)$	Probability of Detection due to attenuation - Describes the probability of detecting an acoustic transmission originating at a range of x meters away from a receiver.
$P_{observation}(T, a, b)$	Percentage of transmission in cell (a, b) which are observable (due to bathymetric obstruction) from cell T . Given a receiver in cell T, and some distribution of animals in the water column of cell (a, b), this value represents the percentage of observable animals in (a, b) from A. Animal distribution is handled by the Vertical Movement Model (Section 3.2.3.3). Section 3.3.3 gives the algorithm for determining bathymetric obstruction and Line of Sight.
$Range(T, i, j)$	Distance from a particular receiver location T to the cell at row i, column j. Computed as the Euclidean distance between the center of a receiver-containing cell T and the center of the cell at row i, column j on a 2D Grid.
R_{opt}	The number of optimal receivers to be placed by the system (excluding projected receivers).
R_{proj}	The number of projected receivers to be placed by the system.
$D_max(p, q)$	The deepest observable depth within a target cell q, relative to the receiver containing cell p. See Section 3.3.3

$Edist_{p,v}$	The Euclidean distance between cells p and v.
$m_{p,q}$	The slope between cells p and q, expressed as the change in bathymetric elevation (of the sea floor) between cells p and q, divided by the Euclidean Distance between cells p and q. See Section 3.3.3
D_{range}	The detection range of the receiver. Assumed to be the same for all receivers in the network. Given as the distance between a receiver and tag at which there is a 5% chance of detecting an acoustic transmission.
$d_{detection}$	The edge size of the Detection Area, equal to $2r + 1$ cells.
f_{supp}	The Suppression Range Factor. A positive real number.
r_{supp}	The Suppression Range. Equal to $f_{supp} * D_{range}$
d_{supp}	The square root of the number of cells in the Suppression Area. The edge dimension of the Suppression Area. Equal to $2 * r_{supp} + 1$ cells.

BIBLIOGRAPHY

[1] Vr2w 69 khz. <http://vemco.com/products/vr2w-69khz/?product-specifications>, 2016. [Online; accessed 30-July-2016].

[2] Vahab Akbarzadeh, Christian Gagné, Marc Parizeau, Meysam Argany, and Mir Abolfazl Mostafavi. Probabilistic sensing model for sensor placement optimization based on line-of-sight coverage. *IEEE Transactions on Instrumentation and Measurement*, 62(2):293–303, 2013.

[3] Perry Barboza. Innovative technology/lab support proposal online form. <https://www.uaf.edu/tab/past-proposals/proposalDetails.xml?id=667>, 2014. [Online; accessed 19-November-2015].

[4] Gregory Burgess, Martin Pedersen, and Kevin Weng. Mande. <https://github.com/gregorylburgess/MANDe/archive/Update.zip>, 2013-2016. [Online; accessed 27-July-2016].

[5] Gregory Burgess, Martin Pedersen, and Kevin Weng. Mande wiki. <https://github.com/gregorylburgess/MANDe/wiki>, 2013-2016. [Online; accessed 27-July-2016].

[6] Rey Farve. Technology and development at the usda forest service, satellite/gps telemetry for monitoring lesser prairie chickens. http://www.fs.fed.us/t-d/programs/im/satellite_gps_telemetry/wildlifetrackingtelementry.htm. [Online; accessed 11-September-2015].

[7] Gretchen J. A. Hansen and Michael L. Jones. The value of information in fishery management. *Fisheries*, 33(7):340–348, 2008.

[8] M. R. Heupel, J. M. Semmens, and a. J. Hobday. Automated acoustic tracking of aquatic animals: Scales, design and deployment of listening station arrays. 57(1):113, 2006.

[9] Andrew Howard, M.J. Mataric, and G.S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, 5:299–308, 2002.

[10] Telonics Inc. Vhf systems for fish (fis). <http://www.telonics.com/products/vhfImplants/vhfFish.php>. [Online; accessed 5-November-2015].

[11] Steven Thomas Kessel, Nigel Edward Hussey, Dale Mitchell Webber, Samuel Harvey Gruber, Joy Michelle Young, Malcolm John Smale, and Aaron Thomas Fisk. Close proximity detection interference with acoustic telemetry: the importance of considering tag power output in low ambient noise environments. *Animal Biotelemetry*, 3(1):1–14, 2015.

[12] Ross A. Maller, Gernot Müller, and Alex Szimayer. *Handbook of Financial Time Series*, chapter Ornstein–Uhlenbeck Processes and Extensions, pages 421–437. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[13] James McNeill. Playtechs: Programming for fun: Raytracing on a grid. `http://playtechs.blogspot.com/2007/03/raytracing-on-grid.html>`, March 2007. [Online; accessed 5-May-2016].

[14] Martin W. Pedersen and Kevin C. Weng. Estimating individual animal movement from observation networks. *Methods in Ecology and Evolution*, 4(10):920–929, 2013.

[15] Martin W. Pedersen and Kevin C. Weng. A state-space model for estimating detailed movements and home range from acoustic receiver data. `<http://orbit.dtu.dk/en/publications/a-statespace-model-for-estimating-detailed-movements-and-home-range-from-acoustic-receiver-data(85d741c7-33ae-47f0-aac0-f0af6b244e3d).html>`, 2013. [Online; accessed 19-April-2016].

[16] S. Poduri and G.S. Sukhatme. Constrained coverage for mobile sensor networks. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 1, 2004.

[17] Collecte Localisation Satellites. How argos works. `<http://www.argos-system.org/web/en/337-how-argos-works.php>`. [Online; accessed 5-November-2015].

[18] Seaturtle.org. Wildlife tracking. `<http://www.wildlifetracking.org/faq.shtml>`. [Online; accessed 5-November-2015].

[19] Anna E Steel, Julia H Coates, Alex R Hearn, and a Peter Klimley. Performance of an ultrasonic telemetry positioning system under varied environmental conditions. *Animal Biotelemetry*, 2(1):15, 2014.

[20] Wikipedia. Animal migration tracking. `<https://en.wikipedia.org/wiki/Animal_migration_tracking#Radio_tracking>`, 2015. [Online; accessed 5-November-2015].

[21] Yuan Zhaohui, Tan Rui, Xing Guoliang, Lu Chenyang, Chen Yixin, and Wang Jianping. Fast sensor placement algorithms for fusion-based target detection. *Proceedings - Real-Time Systems Symposium*, pages 103–112, 2008.