2018-06

ROBUST TIME-VARYING FORMATION CONTROL WITH ADAPTIVE SUBMODULARITY

Wachlin, Noah

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ROBUST TIME-VARYING FORMATION CONTROL WITH ADAPTIVE SUBMODULARITY

by

Noah Wachlin

June 2018

Thesis Advisor:	Douglas P. Horner
Second Reader:	Sean P. Kragelund

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ROBUST TIME-VARYING FORMATION CONTROL WITH ADAPTIVE SUBMODULARITY	**5. FUNDING NUMBERS**
6. AUTHOR(S) Noah Wachlin	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

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

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

An adaptive formation controller is developed to position nodes within a mobile Network Control System (NCS) composed of heterogeneous agents. Each node is represented with distinct capabilities and constraints with regard to communications, sensing, and mobility. Metrics used to quantify network robustness are developed for weighted graphs. Formation control is implemented to position nodes relative to virtual leaders. A utility function that encapsulates the sensing, communications, robustness, and dynamics of the NCS is designed and shown to be submodular. Submodular function maximization is then used to adaptively recompute the optimal formation in simulation. Submodularity is a property of set functions, which guarantees near-optimal performance if a greedy algorithm is used to iteratively select node locations. This effectually reduces the NP-hard combinatorial optimization problem to a polynomial time process. The greedy algorithm is used to adaptively recompute the optimal formation in simulation. This controller reduces the complexity of employing large numbers of autonomous agents in support of competing objectives.

14. SUBJECT TERMS network robustness, network control systems, formation control, adaptive submodularity, combinatorial optimization, heterogeneous, multi-domain, sensor placement	**15. NUMBER OF PAGES** 95
	16. PRICE CODE

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

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ROBUST TIME-VARYING FORMATION CONTROL WITH ADAPTIVE SUBMODULARITY

Noah Wachlin
Ensign, United States Navy
BSE, United States Naval Academy, 2017

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2018

Approved by: Douglas P. Horner
Advisor

Sean P. Kragelund
Second Reader

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

An adaptive formation controller is developed to position nodes within a mobile Network Control System (NCS) composed of heterogeneous agents. Each node is represented with distinct capabilities and constraints with regard to communications, sensing, and mobility. Metrics used to quantify network robustness are developed for weighted graphs. Formation control is implemented to position nodes relative to virtual leaders. A utility function that encapsulates the sensing, communications, robustness, and dynamics of the NCS is designed and shown to be submodular. Submodular function maximization is then used to adaptively recompute the optimal formation in simulation. Submodularity is a property of set functions, which guarantees near-optimal performance if a greedy algorithm is used to iteratively select node locations. This effectually reduces the NP-hard combinatorial optimization problem to a polynomial time process. The greedy algorithm is used to adaptively recompute the optimal formation in simulation. This controller reduces the complexity of employing large numbers of autonomous agents in support of competing objectives.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

CRUSER	Consortium for Robotics and Unmanned Systems Education and Research
DDG	Destroyer
FACTS	Flexible Alternating Current Transmission Device
HVDC	High Voltage Direct Current
IMU	Inertial Measurement Unit
IPB	Intelligence Preparation of the Battlefield
ISR	Intelligence, Surveillance, and Reconnaissance
LTI	Linear Time Invariant
MTX	Multi-Thread Experiment
NCS	Network Control System
NP	Non-deterministic Polynomial-time
NPS	Naval Postgraduate School
NSW	Naval Special Warfare
ROS	Robotic Operating System
SCI	San Clemente Island
SNR	Signal-to-Noise Ratio
UAV	Unmanned Aerial Vehicle
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank my advisor, Dr. Douglas Horner, for his insight, assistance, and direction provided throughout my research. Dr. Horner's guidance and expertise in this subject area were invaluable in refining my research and focusing my efforts to produce a novel and significant contribution to the field.

I would also like to thank Dr. Sean Kragelund for his feedback and assistance on my thesis. His advice was crucial in helping me to clarify my writing and connect disparate concepts. My work with the Hamming supercomputer at NPS would not have been possible without Dr. Kragelund's guidance or Bruce Chiarelli's expertise.

I would also like to express my appreciation to Dr. Marie Wachlin for providing feedback on my writing and providing a perspective from outside the field of Science, Technology, Engineering, and Mathematics (STEM).

Finally, I would like to thank my wife, Charis Wachlin, for her continuous support and being available to discuss a wide variety of technical concepts. Her education in the field of mathematics was an invaluable aid when I lacked the requisite knowledge. Over the past year, she has been a constant source of encouragement.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

In this thesis, we seek to develop a methodology to control a Network Control System (NCS) composed of heterogeneous nodes. The NCS consists of heterogenous agents which include unmanned and manned assets, whereby each node has communications, sensing and mobility capabilities and constraints. Given potentially competing objectives between multiple virtual leaders and tasks, we present a framework for controlling the network nodes as a single system. We will also address metrics for ensuring adequate performance in terms of controllability, observability, and robustness of a Linear Time Invariant (LTI) system. These metrics are incorporated into a utility function that also incorporates the sensing, mobility, and communication constraints. This utility function is shown to be submodular. This property allows us to use the greedy algorithm to solve a combinatorial optimization in polynomial time. This enables the generation of near-optimal formations that are used to dynamically re-position agents relative to the uncontrolled manned platforms represented as virtual leaders.

1.1 Motivation

Unmanned vehicles are becoming increasingly prevalent in all sectors. Many of these unmanned systems are already being used collaboratively in groups. Conceivably, these systems will soon operate in all domains as a single organized system. Each additional unmanned system either provides a novel capability or reduces the cost of an additional operator in an often adverse environment. Despite these benefits, each system sharply increases the size of the necessary support network. When multiple unmanned systems are employed across multiple domains, coordinating those systems becomes increasingly complex, requiring even more manning. Thus, creating an NCS to control these disparate unmanned vehicles as a single system would reduce the complexity and manning necessary to support such groups.

1.2 Problem Statement

This thesis will contribute to research on collaborative robots with the ultimate goal of reducing the manning necessary to control a network of multi-domain unmanned systems. By decreasing the overhead required to coordinate a complex network of mobile sensors, greater flexibility and utility are provided to the operator, and costs can also be reduced. We extend current notions of graph robustness to weighted graphs and develop a submodular utility function for an NCS supporting multiple objectives. The applicability of this research extends beyond the case study examined in this thesis. Examples include search and rescue, surveying areas impacted by natural disasters, fighting wild fires, and oceanographic operations.

1.3 Case Scenario

In November of 2017, the Consortium for Robotics and Unmanned Systems Education and Research (CRUSER) at the Naval Postgraduate School (NPS) conducted a Multi-Thread Experiment (MTX) on San Clemente Island (SCI), CA. The MTX provides a realistic multi-domain scenario to test and increase the autonomy of collaborative unmanned systems. The NCS at the MTX consists of aerial, surface, and undersea assets. These unmanned vehicles and a Navy Destroyer (DDG) operate in support of a Naval Special Warfare (NSW) unit conducting a mission to land on SCI and act on a target. The ScanEagle Unmanned Aerial Vehicle (UAV) shown in Figure 1.1 provides Intelligence, Surveillance, and Reconnaissance (ISR) support with the capability of transmitting live video footage through the network. The SeaFox, a speed-boat sized Unmanned Surface Vehicle (USV) shown in Figure 1.2 provides transportation and limited ISR capabilities with surface search RADAR. The REMUS 100 Unmanned Underwater Vehicle (UUV) pictured in Figure 1.3 is used to map the seafloor with SONAR during Intelligence Preparation of the Battlefield (IPB) before the NSW unit lands on SCI.

We model this NCS as a graph of nodes and links. The unmanned vehicles, NSW unit, target, and support ship (DDG) comprise the nodes of this graph. These nodes are connected by links which represent the sensing and communication relations between these nodes. Once the system model is defined, a controller for the system can be developed. In this thesis we implement *high-level control*, rather than design a control that specifies exact rudder angles or shaft speeds (for example), we design a controller that sends position and velocity

Figure 1.1. The ScanEagle UAV platform.
Photograph by ENS Ben Keegan, NPS.

Figure 1.2. The SeaFox USV platform.
Source: Naval Postgraduate School, https://my.nps.edu/web/cavr/vehicles.

commands to the agents in the NCS. This controller acts as a secondary controller on top of the primary controller onboard the individual agents. Thus, when we use the word controller, we refer to this secondary, high-level controller. This controller must position nodes to maintain the ability to communicate and sense the target and any other threats.

The system works collectively to achieve an objective by driving its state to a desired optimal state for achieving said objective. However, determining this optimal state is often costly and impossible to execute in real-time, so a strategy to approximate this optimal configuration is required rather than attempting to determine it exactly. The development of an optimization function fundamentally seeks to drive a system to achieve an optimality criterion, often by maximizing utility or minimizing cost.

One approach to optimal *coverage* is known as submodular function maximization. If a system is submodular, a near optimal solution can be approximated using a greedy algorithm.

Figure 1.3. The Remus 100 UUV platform.

Source: Naval Postgraduate School, https://my.nps.edu/web/cavr/vehicles.

This method attempts to solve the optimization problem by using the greedy algorithm to iteratively adding elements to a set to maximize the increase in the utility function. This process results in a near optimal solution that can be evaluated in polynomial time [1], [2].

The evaluation of the greedy algorithm requires a utility function to quantify the utility of placing a node in a certain location. Graph theoretic robustness measures described by Ellens and Kooij are extended to quantify the robustness of *weighted* graphs and are incorporated into the utility function [3]. The added benefit of sensing each location is also incorporated into the utility function. The performance of the developed control algorithm is evaluated in simulation and will be tested at future MTX events.

1.4 Overview

In Chapter 2 we review fundamental concepts relating to NCS, optimal coverage, graph theory, and control theory. Then in Chapter 3 we develop and evaluate robustness metrics for weighted networks. We then implement a near-optimal adaptive formation controller with respect to a submodular utility function in Chapter 4. In Chapter 5 we analyze the performance of our controller. We then map our problem to implement adaptive policies using adaptive submodularity in Chapter 6. Finally, we enumerate open areas of research and summarize our results in Chapter 7.

CHAPTER 2:
Background

First, we include a brief literature review of formation control and optimal coverage problems in Sections 2.1 and 2.2, respectively. We then introduce fundamental concepts from graph theory in and control theory, Sections 2.3 and 2.4, respectively.

2.1 Network Control Systems

A *NCS* can be described as a single system composed of multiple independent agents that exchange feedback via networked communications to control the overall state of the system [4]. In recent years, a considerable amount of research has been devoted to NCS. The agents that comprise NCS can either be static or dynamic. An example of a static NCS is provided in [2], which considers a simple model of the European power grid and the placement of High Voltage Direct Current (HVDC) and Flexible Alternating Current Transmission Device (FACTS) nodes to improve the stability of the grid. Our NCS could include static nodes as in [2]; however, we choose to focus on the case where the nodes are mobile (to clarify, these mobile nodes are still permitted to remain stationary).

A mobile NCS differs from its static counterpart in many regards. For instance, the properties surrounding communications cannot be assumed to be time invariant. Communication delays that are time-varying and imperfect information exchange must be taken into account to the design process. As a consequence, it is imperative that the communications and controls are robust and stable across a broad range of operating conditions. In [5], Mesbahi provides a comprehensive overview of topics contained within the domain of NCS, some of which include

- **Consensus:** reaching an agreement on the value of a state variable.
- **Formations:** controlling the agents to assume a specific orientation.
- **Assignments:** tasking agents to (optimally) complete an objective.
- **Flocking:** utilizing simple distributed control laws to emulate *swarming* behaviors observed in many animal species.
- **Coverage:** distributing sensors to observe or estimate an environmental variable.

Flocking is a common means of controlling mobile NCS because control is distributed and therefore scalable to massive networks. Flocking behavior is regulated by three rules identified by Reynolds; cohesion (staying close to other network agents), separation (avoiding collisions with other agents), and alignment (matching the velocity of neighboring agents) [6]. Olfati-Saber provides multiple flocking algorithms using artificial potentials and stability anaylsis in [7]. Li et al also provide a framework for flocking control to optimize communications [8]. Even though flocking behaviors allow the creation of scalable, distributed control, we wish to implement a more deterministic approach. This approach is selected such that node configurations can be explicitly specified. Thus, instead a formation controller is used to position nodes within the network.

In [9], Olfati-Saber, Fax, and Murray outline a common framework for the analysis of consensus and collaborative problems within NCS. Advances in collaborative control (including formation control) are covered in [10]–[15]. Qin, Ma, Shi, and Wang provide details regarding the control of mobile NCS with heterogenous agents in [14]. In [4], Zhang, Han, and Yu provide a survey of communication problems in NCS. We extend this work with formation control to address issues of time-varying formations that are dynamically recomputed to satisfy competing objectives. One such objective is to maintain robust communications. Schuresko and Cortés design and analyze a controller to maintain algebraic connectivity [16], which is a proposed robustness metric based on the graph laplacian (See Subsections 2.3.3 and 3.3.6) [3].

Oh, Park, and Ahn cover recent research on formation control in [17], focusing on *centralized* and *distributed* approaches. In general, distributed approaches are attractive because they are scalable to larger numbers of agents due to the reduced need for communication. This also makes an NCS with distributed control more robust due to the reduced need for communication. However, centralized approaches make reaching a global *consensus* easier. Two implementations of formation control are provided in [18] and [19]. In [19], necessary and sufficient conditions are introduced for controlling *time-varying* formations consisting of *homogenous* agents. We adapt the formation controller described in [19] to drive *heterogeneous* agents to formations that are reassigned to maximize an utility function.

2.2 Optimal Coverage

Numerous techniques have been developed to address coverage optimization problems. The objective of coverage optimization problems is to position nodes of a NCS in an environment, subject to constraints, in order to maximize utility or minimize cost. Given a fixed set of sensor locations subject to area constraints, [20] determines the optimal configuration to reduce the cost of network maintenance using center generalized Voronoi configurations. The work in [21] constructs density functions encoding coverage utility and uses gradient descent methods to optimally position mobile sensing networks. This work was extended in [22] to address optimization for networks where the nodes are subject to communication and anisotropic sensing constraints. In [23], an optimization scheme is presented to minimize the detection time predicted using the Ensemble Cumulative Sum Algorithm. Another approach is discussed in [24], [2], and [1] utilizing submodularity, a property of set functions. Of the many available approaches to combinatorial optimization problems, we utilize *submodular function maximization*.

Submodularity quantifies the property of diminishing returns, i.e., adding an element to a larger set will increase utility less than adding an element to a smaller set would. Krause and Golovin prove that for problems that satisfy this property, a simple greedy algorithm can be used to achieve near optimal performance [1]. Sensor placement is demonstrated in [2] to maximize network controllability and observability. We wish to extend this work by designing robust networks. We will quantify network robustness in terms of metrics discussed in Chapter 3. We discuss submodular function maximization and our implementation of submodular function maximization in greater detail in Section 4.2. In Chapter 6 we discuss the extension of our problem to *adaptive submodularity* to adaptively select optimal policies for our coverage problem. By utilitzing adaptive submodularity, we can use feedback to update our policies to achieve improved performance of the NCS.

2.3 Graph Theory

For a thorough discussion of graph theory and its relation to NCS, see Reference [5]. In general, a graph G is the theoretic representation of a collection of nodes and the connections that exist between them. For instance, a collection of agents and the communication links that exist between them. We define graph $G = (V, E)$ to be the collection of vertices V (nodes) and edges E (links) which connect a subset of V. We describe a singular vertex as

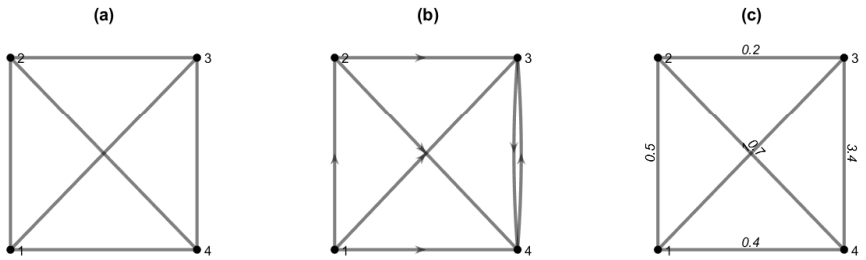

Figure 2.1. Examples of basic graphs.

(a) An undirected graph. (b) A directed graph. (c) A weighted undirected graph. Adapted from [5].

v_i and the edge between *adjacent* vertices v_i and v_j as e_{ij} [5].

In this thesis, all graphs will be a *simple* graph, meaning that at most one edge will exist between each pair of vertices (as opposed to a *multigraph* where more than one edge may exist). We will also omit self-loops (i.e., for $e_{ij}, i \neq j$). Figure 2.1(a) shows an example of an *undirected* graph. If the edges are directed, it is accordingly referred to as a *directed* graph or *digraph*, as in Figure 2.1(b). We also make use of the concept of *weighted graphs* $G = (V, E, w)$, where w is the set of weights corresponding to the edges E as shown in Figure 2.1(c). For example, an edge weight could quantify the strength of the communication link between two nodes.

2.3.1 Adjacency Matrix

Although it is intuitive to represent these networks graphically, it is often more useful to represent the graph using matrices. The *adjacency matrix* $\mathbf{A}(G)$ encodes the adjacency of vertices i and j where

$$A_{ij}(G) = \begin{cases} w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}. \tag{2.1}$$

If the graph is undirected, the adjacency matrix is always symmetric. If the graph is unweighted, $w_{ij} = 1$ [5]. For example, in the case of the digraph in Figure 2.1(b), the adjacency matrix is

$$\mathbf{A}(G) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

and the weighted adjacency matrix of the graph in Figure 2.1(c) is

$$\mathbf{A}(G) = \begin{bmatrix} 0.0 & 0.5 & 1.0 & 0.4 \\ 0.5 & 0.0 & 0.2 & 0.7 \\ 1.0 & 0.2 & 0.0 & 3.4 \\ 0.4 & 0.7 & 3.4 & 0.0 \end{bmatrix}.$$

2.3.2 Degree Matrix

For an undirected graph, the *degree* of a vertex $\delta(v_i)$ quantifies the number of vertices adjacent to the vertex v_i. The degree can also be seen as the number of edges connected to vertex v_i. A graph is said to be *complete* if the degree of every vertex is $n - 1$ [5]. The degrees of the vertices in the graph in Figure 2.1(a) are

$$\delta(v_1) = \delta(v_2) = \delta(v_3) = \delta(v_4) = 3$$

These can be assembled in a diagonal *degree matrix* $\mathbf{\Delta}(G)$. For an undirected graph G with n vertices [5],

$$\mathbf{\Delta}(G) = \begin{bmatrix} \delta(v_1) & 0 & \dots & 0 \\ 0 & \delta(v_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \delta(v_n) \end{bmatrix}. \tag{2.2}$$

For directed graphs, *indegree* and *outdegree* are used to differentiate between inward directed edges that terminate at vertex v_i and outward directed edges that originate at vertex v_i [5],

2.3.3 The Laplacian

The Laplacian is important as it relates to the controllability of our NCS [5]. The graph Laplacian $\mathbf{L}(G)$ is simply defined as the difference of the degree matrix $\mathbf{\Delta}(G)$ and the adjacency matrix $\mathbf{A}(G)$ (i.e., $\mathbf{L}(G) = \mathbf{\Delta}(G) - \mathbf{A}(G)$) [5]. Thus, the Laplacian takes the form

$$L_{ij}(G) = \begin{cases} \delta_i & \text{if } i = j \\ -1 & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

The Laplacian of the graph in Figure 2.1(a) is

$$\mathbf{L}(G) = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix},$$

The Laplacian for undirected graphs has a few special properties; it is symmetric, positive semidefinite, and the rows sum to zero. Thus, its eigenvalues are real, non-negative, and the smallest one is zero [3]. These eigenvalues ($\boldsymbol{\lambda}$) are commonly ordered such that

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n. \tag{2.4}$$

For directed graphs we will use the outdegree to compute the Laplacian (note, the properties discussed in the previous paragraph do not hold for digraphs). Thus for Figure 2.1(b), the Laplacian is

$$\mathbf{L}(G) = \begin{bmatrix} 3 & -1 & -1 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix},$$

2.4 Control Theory

Modern control theory provides many tools to analyze and design controllers for a wide variety of enclosed systems. By modeling a system in state-space representation, we can determine the controllability and observability of nonlinear and linear systems. Liu and Barabási provide extensions to NCS in [25].

2.4.1 State-Space Representation

Any dynamic system with n states, p inputs, and q outputs, can be represented in the most general state-space representation as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)), \tag{2.5}$$

$$\mathbf{y}(t) = \mathbf{h}(t, \mathbf{x}(t), \mathbf{u}(t)), \tag{2.6}$$

where the $\mathbf{x}(t) \in \mathbb{R}^n$ represents the state of the system at time t, $\mathbf{u}(t) \in \mathbb{R}^p$ represents the inputs to the system, and $\mathbf{y}(t) \in \mathbb{R}^q$ represents measurements. For instance, $\mathbf{x}(t)$ could represent the pose of a UAV, $\mathbf{u}(t)$ the commands sent to its control surfaces and propulsion system, and $\mathbf{y}(t)$ the measurements generated by the onboard Inertial Measurement Unit (IMU).

If $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are linear (or approximately linear around an operating point), the state-space representation becomes

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t), \tag{2.7}$$

$$\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t), \tag{2.8}$$

where $\mathbf{A}(t) \in \mathbb{R}^{n \times n}$ is the state matrix, $\mathbf{B}(t) \in \mathbb{R}^{n \times p}$ is the input matrix, $\mathbf{C}(t) \in \mathbb{R}^{q \times n}$ is the output matrix, and $\mathbf{D}(t) \in \mathbb{R}^{q \times p}$ is the feedforward matrix. If the system is time-invariant, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ lose their dependence on time, in which case the system is referred to as an LTI system.

Previously, state-space representation and analysis have been limited to singular systems. However, as computational power and our ability to map more complex networks of systems have grown, the problem of controlling networks has become more tractable. In [25], Liu and Barabási provide a thorough coverage of control theory as applied to these complex networks. For a NCS, the weighted adjacency matrix is used as the state matrix $\mathbf{A}(t)$ for the

system. In this way, the state matrix represents how the nodes are connected. The amount of traffic passing through node i is denoted by the state variable $x_i(t)$. The populated cells in the input matrix $\mathbf{B}(t)$ indicate which nodes are directly controlled by the input signal $\mathbf{u}(t)$ and the influence the input has on those nodes [25].

2.4.2 Controllability

A system is *controllable* if, starting from any initial condition, it can be driven in a finite amount of time to a desired final state [26]. Several tools exist to determine the controllability of LTI systems. For a system with n states, the controllability rank criterion guarantees that if the controllability matrix has full rank, i.e. $\text{rank}(\mathbf{M}_c) = n$, the system is controllable, where the controllability matrix is

$$\mathbf{M}_c = [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}]. \tag{2.9}$$

This metric is particularly binary, simply indicated that if the system is controllable or not. However, to gain more resolution on the controllability of a system the *contollability Grammian* $\mathbf{W}_c(t)$ can be used. The symmetric positive definite controllability Grammian is related to the amount of input energy required to reach a given state from the initial condition [2]. The finite horizon controllability Grammian is defined

$$\mathbf{W}_c(t) = \int_0^t e^{\mathbf{A}\tau} B B^T e^{\mathbf{A}^T\tau} d\tau \in \mathbb{R}^{nxn}. \tag{2.10}$$

The finite horizon Gramian can be used to analyze both stable and unstable systems, however it is harder to evaluate than the infinite horizon Gramian $\left(\lim_{t\to\infty} \mathbf{W}_c(t)\right)$. The infinite horizon Gramian can instead be determined by solving a Lyapunov equation

$$\mathbf{AW}_c + \mathbf{W}_c\mathbf{A}^T + \mathbf{BB}^T = 0. \tag{2.11}$$

This advantage becomes greater as the size of the network increases [2].

Theories of structural controllability have also been developed for NCS. In [25] and [27], the theories of structural controllability developed by Lin are covered thoroughly to prove controllability for state matrices where the structure is known but the value of its nonzero elements are not precisely known or vary with time. With concepts from structural control-

lability, the minimum number of "driver nodes" (nodes that are directly controlled) can be more easily identified in order to ensure controllability, rather than conducting an exhaustive brute force search (i.e., to determine the structure of $\mathbf{B}(t)$) [27].

2.4.3 Observability

A system is *observable* if, given a particular subset of state measurements, it is possible to estimate all states in the system. Analogous tools exist to test for observability in a similar manner as controllability. For an LTI system, the observability rank criterion states if the observability matrix

$$\mathbf{M}_o = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{2.12}$$

has full rank (i.e., $\text{rank}(\mathbf{M}_o) = n$), then the system is observable. Numerous other tests exist to determine the observability of a system, including an observability Gramian; however we will not make use of them in this thesis [25].

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Robust Networks

Numerous graph theoretic metrics have been proposed to quantify the robustness of graphs. As the physical architecture of NCS lend themselves naturally to being represented by graphs, it is natural that we extend these robustness metrics to our NCS. Ellens and Kooij provide a comprehensive spread of proposed metrics to quantify network robustness in [3]. Yang et al. use four of these metrics to demonstrate the robust growth of networks in simulation. [28]. In this chapter, we describe desirable characteristics of a robustness metric (Section 3.1). We then describe proposed robustness metrics (Section 3.3) and provide results showing the evolution of these metrics for the growth of a generalized network (Section 3.4). (Note: We use "network" and "graph" (the representation of the network) interchangeably).

3.1 Desired Characteristics

In order to better quantify network robustness, we identify subdivisions of robustness and attempt to increase the amount of information encapsulated in our robustness metric. In relation to networks, robustness is a vague and imprecisely defined term. Thus, we identify four more intuitive factors which affect network robustness. We connect these four factors to four subdivisions of robustness that we define as resiliency, toughness, flexibility, and strength. We then connect these subdivisions to metrics defined in Section 3.3. In our development of a single robustness metric we require that it quantify these four factors:

- Edge addition and deletion (resiliency).
- Node addition and deletion (toughness).
- Network flexibility (flexibility).
- Link strength (strength).

We refer to the ability of the NCS to continue functioning after the loss of an edge as the *resiliency* of the network. The loss of an edge is equivalent to when direct communication between adjacent nodes is severed. Of the proposed measures, the edge connectivity best quantifies network resiliency (Subsection 3.3.1).

We describe *toughness* as the ability of the NCS to continue functioning after the loss of a node. Toughness is largely dependent on the topology of the graph. Barabási describes two major network topologies; pseudorandom and scale-free [29]. In a pseudorandom graph, toughness is relatively low because the number of nodes necessary to disconnect the graph is relatively low, whereas scale-free networks are resistant to the removal of a large number of nodes. Thus, the toughness of scale-free networks is high, since it can handle a large number of random node removals. However, the *vulnerability* of scale-free networks is also high because a targeted attack on a hub can disconnect the network [29]. The clustering coefficient and number of spanning trees best quantify toughness (Subsections 3.3.4 and 3.3.7 respectively).

Flexibility indicates the ability of a network to adjust its topology to accommodate changing objectives. Intuitively a more flexible graph is more robust because of its ability to compensate for changing conditions. Of the metrics proposed in the literature, the effective graph resistance seems to best quantify network flexibility, however this is still open for consideration (Subsection 3.3.8).

The *strength* of the network indicates the communication capacity of the network. Clearly, the network strength is directly related to the strength of the links between individual nodes. This link strength is represented as the edge weight w_{ij}. In the literature, discussions of network robustness have not considered weighted graphs. However, we find that without considering network strength, any robustness metric does not provide sufficient resolution for cases when the network topology changes without the loss or addition of edges and vertices (i.e., when the edge weights change). For this reason, we devote Section 3.2 to describing various edge weight functions and adapt metrics developed for unweighted graphs to be used with weighted graphs in Section 3.3.

3.2 Edge Weight Functions

We desire our robustness metric to have a high resolution to reflect small topological changes in the network that do not include the addition or removal of nodes in links. For this reason, we propose (when possible) to use weighted graphs. Weighted graphs allow us to capture more information about the physical state of the network, thereby providing a

better indication of robustness. In all cases, we normalize these weights such that they are bounded from zero to one.

For example, the edge weight could encapsulate the physical range r_{ij} between two adjacent nodes v_i and v_j. We assume a maximum communication range $r_{\max}$. The edge weight w_{ij} then decreases *proportionally* to the range up to $r_{\max}$

$$w_{ij}(r_{ij}) = \begin{cases} 1 - 0.9\frac{r_{ij}}{r_{\max}} & \text{if } r_{ij} \leq r_{\max} \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases}. \tag{3.1}$$

Note that we scale this proportionally decreasing edge weight such that the minimum value attained before disconnecting is $w_{ij} = 0.1$. For each edge weight function we enforce that the minimum weight is no less than 0.1 to differentiate from when the edge does not exist, i.e., $w_{ij} = 0$. In Equation 3.1, the network becomes less robust as the edge weight decreases. This distribution is shown in Figure 3.1(a) where the maximum range is one meter. This maximum range can be increased to whatever value suits the reader; for instance, the maximum communications range of a radio system.

Perhaps a more intuitive formulation for a NCS would be to quantify the edge weight with a function that encapsulates the communication strength. For instance, a function approximating the shape of the path loss could be expressed as an *inverse exponential* with the range scaled by the maximum communications range.

$$w_{ij}(r_{ij}) = \begin{cases} e^{\tau r_{ij}} & \text{if } r_{ij} \leq r_{\max} \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases}, \tag{3.2}$$

where τ determines the decay rate of the weighting function. This distribution is shown in Figure 3.1(b) where $\tau = 2.5$ and $r_{\max} = 1.0$ meter. This function could also represent the probability that a message will be received. Additionally, rather than being a function of range, these edge weight functions could instead be functions of the bandwidth of the communication channel or the received Signal-to-Noise Ratio (SNR).

In some applications we may be less interested in making the edge weight proportional to the communication strenght. For instance, if we know that we have a sufficient communication signal up to a certain range. This type of distribution is shown in Figure 3.1(c), where

sufficient communications exist up to range r_{suf} at which point the communication quality degrades and the link is broken at $r_{\max}$. This distribution is given by

$$w_{ij}(r_{ij}) = \begin{cases} 1 & \text{if } r_{ij} \leq r_{\text{suf}} \\ 0.1 + 0.45\left(1 + \cos\left(\pi \frac{r_{ij} - r_{\text{suf}}}{r_{\max} - r_{\text{suf}}}\right)\right) & \text{if } r_{\text{suf}} \leq r_{ij} \leq r_{\max} \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases}. \tag{3.3}$$

For the distribution shown in Figure 3.1(c), $r_{\text{suf}} = 0.5\text{m}$ and $r_{\max} = 1\text{m}$. Although this is formulated as a function of range, it could be easily adjusted to be a function of SNR, bandwidth, or any other environmental variable. A similar function could be used to enforce a separation distance $r_{\min}$ between nodes. This function maximizes the edge weight at the midpoint between $r_{\min}$ and the maximum interaction range $r_{\max}$

$$w_{ij}(r_{ij}) = \begin{cases} 0.1 & \text{if } r_{ij} \leq r_{\min} \\ 0.1 + 0.45\left(1 + \cos\left(\pi \frac{r_{ij} - r_{\min}}{r_{\max} - r_{\min}}\right)\right) & \text{if } r_{\min} \leq r_{ij} \leq r_{\max} \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases}, \tag{3.4}$$

where $r_{\text{mid}} = r_{\min} + \frac{1}{2}(r_{\max} - r_{\min})$. This distribution is shown in Figure 3.1(d) where $r_{\min} = 0.5\text{m}$ and and $r_{\max} = 1.0\text{m}$.

Alternatively, rather than having the maximum edge weight represent the most robust link, we could have the minimum edge weight represent the most robust link. For instance, the edge weight could quantify the chance that a communication packet is not received. Thus, a lower edge weight (and accordingly lower probability) indicates a more robust link. Using this edge weight formulation lends itself better to certain robustness metrics as shown in Section 3.4. An edge weight distribution that increases proportionally to the range between nodes is

$$w_{ij}(r_{ij}) = \begin{cases} 0.1 + 0.9\frac{r_{ij}}{r_{\max}} & \text{if } r_{ij} \leq r_{\max} \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases}. \tag{3.5}$$

This distribution is shown in Figure 3.2(a). The inverse exponential edge weight can also

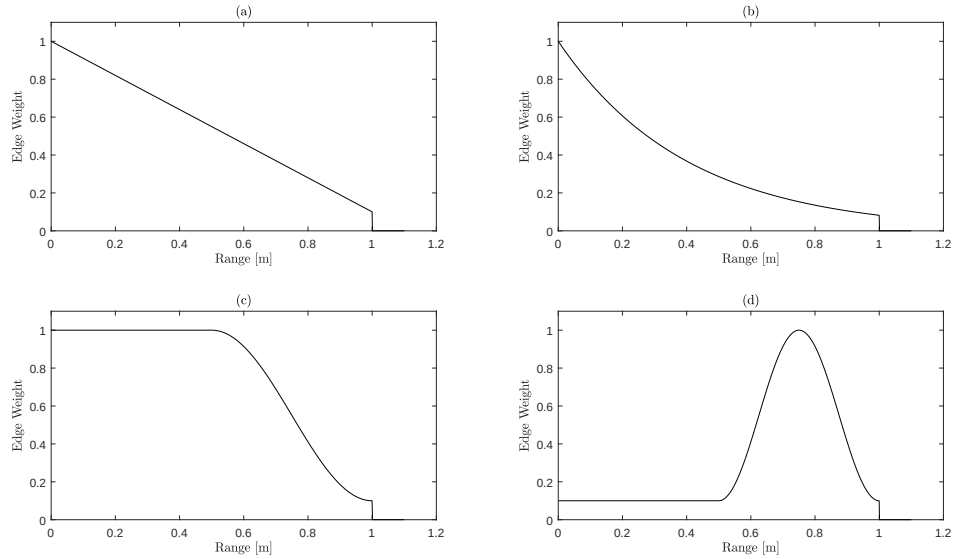

Figure 3.1. Decreasing edge weight distributions.

Distributions are shown where decreasing edge weight indicates decreasing robustness. These are shown for ranges from zero to 1.1 meters, where the edge weight function (a) is porportional to the range as in Eq. 3.1, (b) an inverse exponential function of the range as in Eq. 3.2, (c) smoothly varies from one to zero as in Eq. 3.3, and (d) is maximized at r_{mid} as in Eq. 3.4.

be modified as shown in Figure 3.2(b) and computed as

$$w_{ij}(r_{ij}) = \begin{cases} 1.0 - 0.9 * e^{\tau r_{ij}} & \text{if } r_{ij} \leq r_{\max} \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases}. \tag{3.6}$$

Similarly, the two sinusoidal edge weight distributions are

$$w_{ij}(r_{ij}) = \begin{cases} 0.1 & \text{if } r_{ij} \leq r_{\text{suf}} \\ 0.1 + 0.45\left(1 - \cos\left(\pi \frac{r_{ij} - r_{\text{suf}}}{r_{\max} - r_{\text{suf}}}\right)\right) & \text{if } r_{\text{suf}} \leq r_{ij} \leq r_{\max}, \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases} \tag{3.7}$$

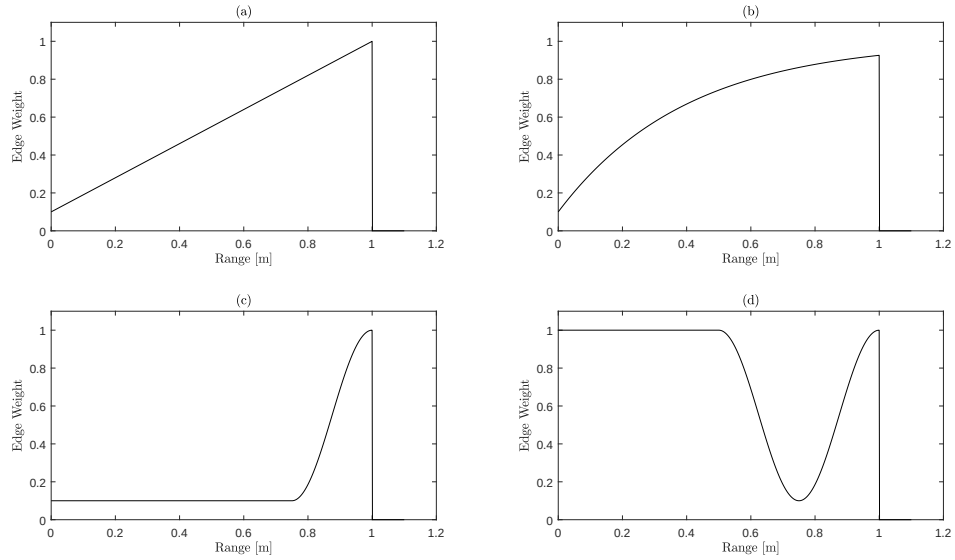

Figure 3.2. Increasing edge weight distributions.

Distributions are shown where increasing edge weight indicates decreasing robustness. These are shown for ranges from zero to 1.1 meters, where the edge weight function (a) is porportional to the range as in Eq. 3.5, (b) an inverse exponential function of the range as in Eq. 3.6, (c) smoothly varies from one tenth to one as in Eq. 3.7, and (d) is minimized at r_{mid} as in Eq. 3.8.

and

$$w_{ij}(r_{ij}) = \begin{cases} 1 & \text{if } r_{ij} \leq r_{\min} \\ 0.1 + 0.45\left(1 - \cos\left(\pi \frac{r_{ij} - r_{\min}}{r_{\max} - r_{\min}}\right)\right) & \text{if } r_{\min} \leq r_{ij} \leq r_{\max} \\ 0 & \text{if } r_{ij} > r_{\max} \end{cases}. \tag{3.8}$$

These distributions are shown in Figure 3.2(c) and (d) respectively.

Each of the eight edge weight distributions described in this section can be placed into one of two categories: distributions in which a high edge weight indicates stronger communication (e.g., edge weight decreases as a function of range) or distributions in which a lower edge weight indicates stronger communications (e.g., edge weight decreases as a function of range). We examined each of these edge weight distributions for each the robustness metrics identified in Section 3.3 to be compatible with weighted graphs. We provide plots of these metrics for the two edge weight distributions that either increase or decrease proportionally to the range. This allows us to determine which type of edge weight distribution results in

a more intuitive value of the robustness metric. Then in implementation, we select an edge weight distribution that better quantifies the physical interaction between the agents in the NCS.

3.3 Graph Theoretic Robustness Measures

In this section we discuss the metrics proposed by Ellens and Kooij in [3] and by Yang et al. in [28]. We also indicate how each metric relates to graph robustness. We expand upon these authors' work by incorporating *weighted* graphs into the robustness measures where possible. In this section, however, we focus solely on a general *undirected* graph with n nodes and m edges. We do not consider the extension of these metrics to digraphs, which is an open area of research. Throughout this section we indicate the number of nodes with n and the number of edges with m.

3.3.1 Connectivity

Connectivity κ is a binary metric which indicates if a graph is connected ($\kappa = 1$) or disconnected ($\kappa = 0$). If a graph is connected, then there exists a path between any two vertices v_i and v_j. If the algebraic connectivity (the second eigenvalue λ_2 of the Laplacian $\mathbf{L}(G)$) is greater than zero, then the $\kappa = 1$ [3].

Vertex connectivity (κ_v) is equivalent to the minimum number of vertices necessary to remove in order to disconnect the graph [3].

Edge connectivity (κ_e) is defined as the minimum number of edges necessary to remove in order to disconnect the graph [3].

Algorithms used to compute the vertex and edge connectivity are provided in [30]. However, the computation of these metrics is NP-hard and becomes increasingly computationally expensive as the size of the graph grows [30]. For this reason, we do not consider them to be good candidates to be robustness metrics. One may note however, $\kappa \leq \kappa_v \leq \kappa_e \leq \delta_{\min} \leq n - 1$, where $\delta_{\min} = \min(\mathbf{\Delta}(G))$, the minimum degree of the graph G. Intuitively, the NCS becomes more robust as each of these quantities increase. Although it is not proposed in [3] as a measure of robustness, the computation of $\delta_{\min}$ is much simpler than that of edge and

vertex connectivity, so $\delta_{\min}$ could be used as a *lazy evaluation* of the edge connectivity to speed the evaluation of a greedy algorithm used as part of an optimization scheme.

3.3.2 Distance Based Measures

We call the length (sum of the edge weights) of the shortest path between vertices v_i and v_j, the distance d_{ij}. Note that v_i and v_j need not be adjacent and that d_{ij} may not be a number if the graph is disconnected ($\kappa = 1$).

The *diameter* of the graph is then defined in [3] as

$$d_{max} = \max d_{ij}, \tag{3.9}$$

and the *average distance* is defined as

$$\bar{d} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}. \tag{3.10}$$

For an unweighted graph, increasing distance indicates decreasing robustness as a larger distance indicates information must travel through more nodes en route from v_i to v_j. However, if weighted edge distributions are used as described previously, a larger distance indicates a stronger communication link.

The *efficiency* is also proposed in [3] as the average of the inverse of the distances

$$E = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{d_{ij}}. \tag{3.11}$$

For unweighted graphs, increasing efficiency indicates increasing robustness since the reciprocal of distance is used. For weighted graphs, the opposite is true.

One shortcoming of distance based measures identified in [3] is the fact that alternate paths between v_i and v_j are not considered. Naturally one would assume a network with multiple paths linking each node is more resilient to the loss of of an edge and has greater toughness to withstand node failures. However, this is not captured by these metrics.

3.3.3 Betweenness

The *betweenness* indicates the number of shortest paths *passing through* a vertex or an edge x between vertices v_i and v_j,

$$b_x = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{n_{ij}(x)}{n_{ij}}, \tag{3.12}$$

where n_{ij} is the number of shortest paths between v_i and v_j and $n_{ij}(x)$ is the number of shortest paths passing through x between v_i and v_j [3]. The betweenness quantifies the importance of an edge or vertex x. If a single vertex or edge has a significantly higher betweenness, it can indicate a vulnerability in the network. Ellens and Kooij show that for *unweighted* graphs, the *average vertex betweenness* $\bar{b}_v$ and *average edge betweenness* $\bar{b}_v$ are functions of the average distance

$$\bar{b}_v = \frac{1}{2}(n-1)(\bar{d}+1) \tag{3.13}$$

$$\bar{b}_e = \frac{n(n-1)}{2m}\bar{d}. \tag{3.14}$$

Note these relationships do not hold for *weighted* graphs. Since Equations 3.13 and 3.14 are only valid for unweighted graphs, here the average distance is equivalent to the average number of edges along the shortest path between all nodes v_i and v_j, instead of the sum of the edge weights. Robustness increases as both of these quantities increase.

3.3.4 Clustering Coefficient

In a social network, people are represented as nodes and friendships are indicated by edges. The clustering coefficient originally used in this context to compute the probability that two people (v_i and v_j) are friends ($\mathrm{P}(e_{ij} = 1)$), given that v_i is friends with v_k (edge e_{ik} exists) and v_j is also friends with v_k. Thus, the *clustering coefficient* C captures the probability that the triangle formed by v_i, v_j, and v_k exists. Robustness increases with the clustering coefficient, which can be computed

$$C = \frac{1}{n} \sum_{i \in V : \delta_i > 1} \frac{1}{\delta_i(\delta_i - 1)} [\mathbf{A}^3(G)]_{ii}, \tag{3.15}$$

where $\mathbf{A}(G)$ is the adjacency matrix and δ_i is the degree of node i [3]. In Section 3.4, we examine the effect of using the weighted adjacency matrix instead of the unweighted adjacency matrix.

3.3.5 Reliability Polynomials

Given the probability p that an edge e exists, the reliability polynomial of a network (Rel(G)), is then the probability that the network is connected. This probability is defined as

$$\text{Rel}(G) = \sum_{i=0}^{m} F_i (1-p)^i p^{m-i}, \tag{3.16}$$

where F_i is the number of connected subgraphs of G if i edges are removed [3]. This quantity lends itself naturally to using the edge weights as the probability p which must be selected. However, as the size of the graph grows, F_i becomes exponentially more expensive to compute. For this reason, for networks we approximate F_i as

$$\hat{F}_i = \frac{m}{i} \tag{3.17}$$

as suggested in [31]. Robustness increases as the evaluation of the reliability polynomial increases.

3.3.6 Algebraic Connectivity

The *algebraic connectivity* is defined as the second smallest eigenvalue (λ_2) of the Laplacian [3]. Unfortunately, this measure is not strictly increasing as edges are added, so it is hard to correlate to robustness. In general, as λ_2 increases, so does robustness. In unweighted networks, the algebraic connectivity is never greater than the vertex connectivity. Thus we may arrange some metrics in order of increasing magnitude

$$0 \leq \lambda_2 \leq \kappa_v \leq \kappa_e \leq \delta_{\min} \leq n-1. \tag{3.18}$$

This metric is far less intuitive, so we attempt to correlate its behavior through by plotting it as function of the edge weight in Section 3.4.

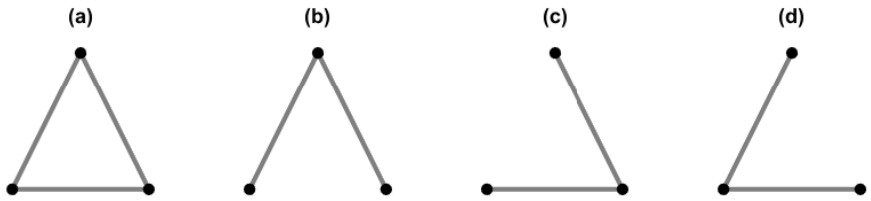

Figure 3.3. Example spanning trees.

A graph G is shown in (a) for which $\zeta = 3$. These three spanning trees are shown in (b), (c), and (d). Adapted from [5].

3.3.7 Number of Spanning Trees

Given graph G with n vertices, a spanning tree is a subgraph of G containing $n - 1$ edges that also forms a tree (i.e., there are no cycles) [3]. This is illustrated for a simple graph in Figure 3.3. The *number of spanning trees* ζ is proportional to the product of the eigenvalues (λ_i) of the unweighted Laplacian:

$$\zeta = \frac{1}{n} \prod_{i=2}^{n} \lambda_i. \tag{3.19}$$

A proof is referenced in [3], and this can be easily verified for the graph in Figure 3.3(a); The Laplacian of of the graph is

$$\mathbf{L}(G) = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix},$$

whose eigenvalues are $\boldsymbol{\lambda} = [0\ 3\ 3]^T$. The number of spanning trees is then

$$\zeta = \frac{1}{3}(3)(3) = 3,$$

as shown graphically in Figure 3.3. Robustness intuitively increases with the number of spanning trees since it indicates a network with more redundant edges.

3.3.8 Effective Graph Resistance

The *effective graph resistance* (R) is inspired by first modeling the graph as a circuit with resistors between each node, each having a resistance of one. The effective graph resistance

is then the the sum of effective resistances between each pair of vertices, as determined by using Kirchoff's laws [3]. For unweighted networks, it has been shown that the effective resistance is incredibly equivalent to

$$R = n \sum_{i=2}^{n} \frac{1}{\lambda_i}. \tag{3.20}$$

This metric accounts for all paths between each vertex pair and their path lengths. For this reason, it is a very attractive metric that provides a strong indication of network robustness. For an unweighted graph, robustness increases as the effective graph resistance decreases. In Section 3.4 we provide results for the computation of the effective resistance when the eigenvalues of the weighted graph Laplacian are used instead.

3.4 Results

We use the graphs shown in Figure 3.4 to compare the proposed robustness metrics. The graphs are ordered from left to right in what we would intuit as increasing robustness. The proposed metrics are tabulated in Table 3.1 for these five graphs where the weight on all edges is one (this is equivalent to evaluating the measures on an unweighted graph).

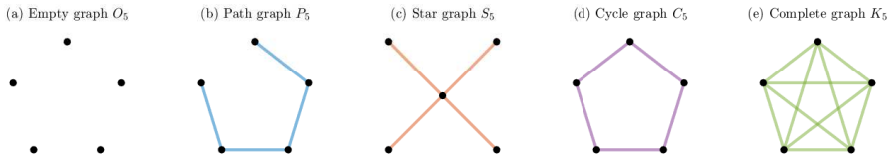

Figure 3.4. Sample graphs.

Sample graphs are shown to be used to compute the proposed robustness metrics. Graphs are arranged from left to right in increasing order of expected robustness.

We are particularly interested in the metrics where the weights affect the value of the metric in addition to the structure: the diameter $d_{\max}$, average distance $\bar{d}$, efficiency E, clustering coefficient C, reliability polynomial $\mathrm{Rel}(G)$, algebraic connectivity λ_2, and effective resistance R. We show these metrics in Figures 3.5 through 3.11 plotted against the separation range between adjacent nodes. In each figure, the left panel uses the decreasing weight distribution described by Equation 3.1 and the right panel uses the increasing edge weight distribution described by Equation 3.5. For the sake of simplicity, the maximum commu-

Table 3.1. Robustness metrics unweighted graphs.

	κ	κ_v	κ_e	$\delta_{\min}$	$d_{\max}$	$\bar{d}$	E	$b_{e,\max}$	$\bar{b}_e$	$\bar{b}_v$	C	λ_2	ζ	R
O_5	0	0	0	0	∞	∞	0.00	-	-	-	0.00	0.00	0	∞
P_5	1	1	1	1	4.00	2.00	0.64	4	6	5	0.00	0.38	1	4.00
S_5	1	1	1	1	2.00	1.60	0.70	6	5	4	0.00	1.00	1	3.20
C_5	1	2	2	2	2.00	1.50	0.75	1	5	3	0.00	1.38	5	2.00
K_5	1	-	5	4	1.00	1.00	1.00	0	4	1	1.00	5.00	125	0.80

The values of proposed robustness metrics for the graphs in Figure 3.4 where the edge weight, $w_{ij} = 1$.

nications range $r_{\max}$ is set to one meter and each are plotted for $r_{i,j} = [0.0, 1.1]$. These plots aid in determining which type of distribution is best suited to the computation of each metric. The results of this analysis are summarized in Table 3.2.

The diameter is inversely proportional to robustness. In Figure 3.5, complete graph K_5 always has the smallest diameter as one would expect. In general, as the range between nodes grows larger we want the metric to reflect the decreasing robustness. Thus, a decreasing edge weight distribution should be used as shown in the left panel (Figure 3.5(a)). One may note than when the edge weight goes to zero beyond $r_{\max}$, the diameter goes to infinity.

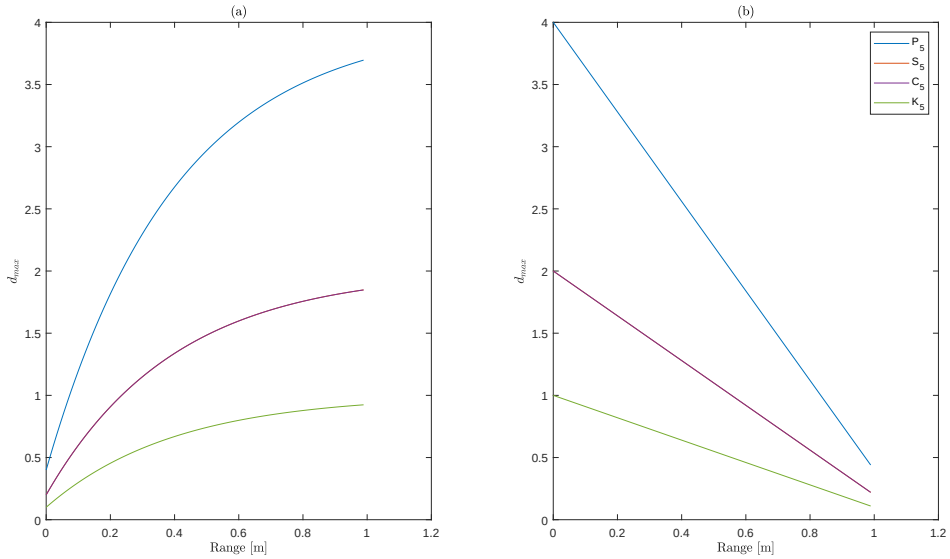

Figure 3.5. The diameter $d_{\max}$ vs. range.

The *diameter* $d_{\max}$ is shown relative to the range for a path graph, a star graph, a cycle graph, and a complete graph for (a) an increasing edge weight distribution (Eq. 3.1) and (b) a decreasing edge weight distibution (Eq. 3.5) that are linear functions of the range. (Note: the line for S_5 is directly behind P_5.)

The average distance is also inversely proportional to robustness. In Figure 3.6, the lines representing the metric for each line are ordered as expected; bottom to top in order of decreasing robustness. By using a decreasing edge weight distribution, as shown in the left panel (Figure 3.6(a)), the average distance also reflects the decreasing robustness associated with increasing the distance between nodes. One may note than when the edge weight goes to zero beyond $r_{\max}$, the diameter goes to infinity.

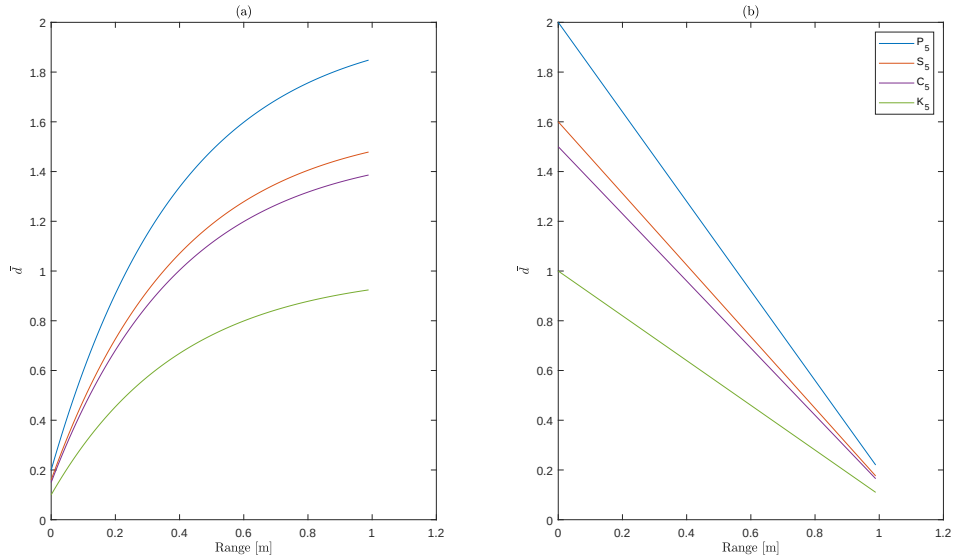

Figure 3.6. The average distance $\bar{d}$ vs. range.

The *average distance* $\bar{d}$ is shown relative to the range for a path graph, a star graph, a cycle graph, and a complete graph (a) an increasing edge weight distribution (Eq. 3.1) and (b) a decreasing edge weight distibution (Eq. 3.5) that are linear functions of the range.

Graph efficiency increases with robustness. Thus, we wish to maximize the efficiency when the range is the smallest. An increasing edge weight distribution accomplishes this, as shown in Figure 3.7(a).

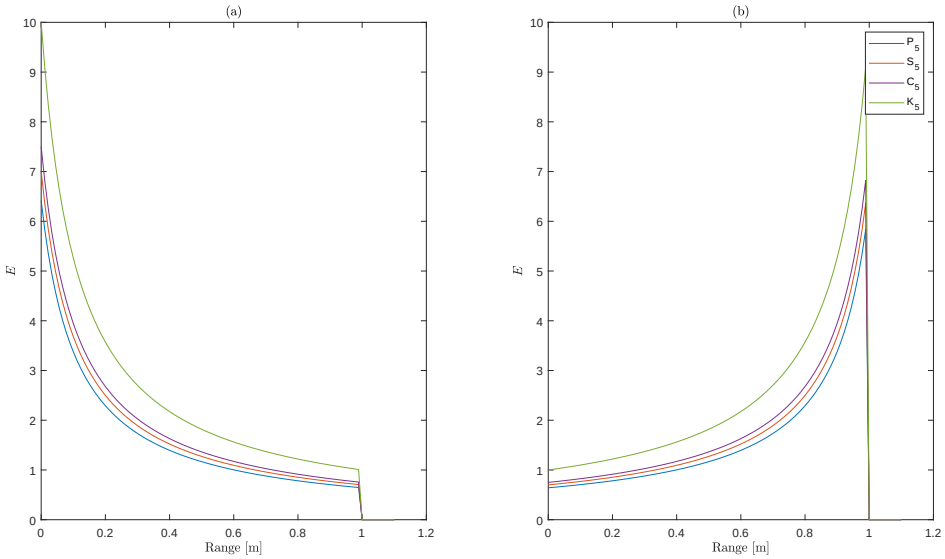

Figure 3.7. The efficiency E vs. range.

The *efficiency* E is shown relative to the range for a path graph, a star graph, a cycle graph, and a complete graph for (a) an increasing edge weight distribution (Eq. 3.1) and (b) a decreasing edge weight distibution (Eq. 3.5) that are linear functions of the range.

Figure 3.8 exposes the weaknesses of using the clustering coefficient to quantify the robustness of weakly connected graphs; it does not distinguish between the graphs P_5, S_5, and C_5 since none of them contain redundant edges like K_5. However, if used, we wish to maximize C, thus a decreasing edge weight distribution is best for reflecting the decreasing robustness as the range between nodes decreases (see Figure 3.8(b)).

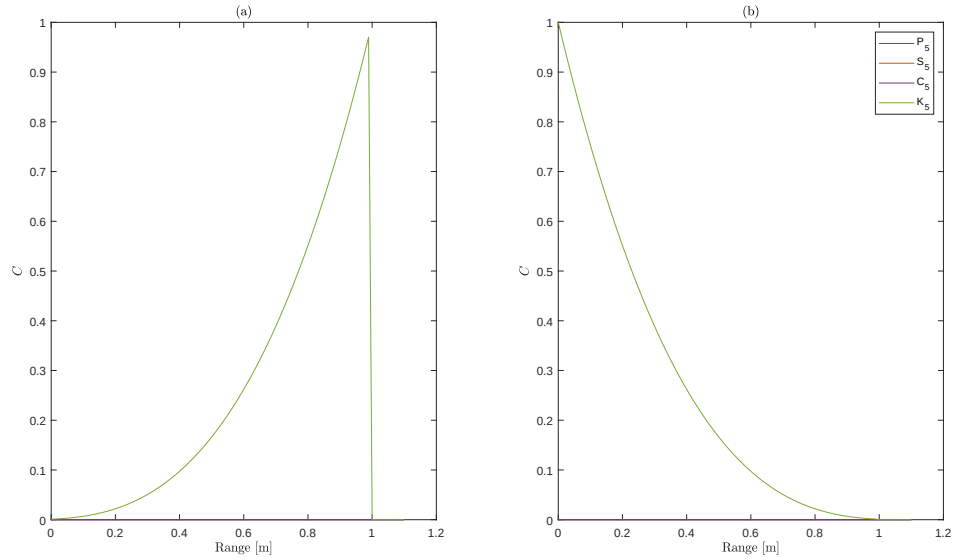

Figure 3.8. The clustering coefficient C vs. range.

The *clustering coefficient* C is shown relative to the range for a path graph, a star graph, a cycle graph, and a complete graph for (a) an increasing edge weight distribution (Eq. 3.1) and (b) a decreasing edge weight distibution (Eq. 3.5) that are linear functions of the range.

As seen in Figure 3.9, the reliability polynomial does not appear to be directly correlated to the edge weight. Although we thought the reliability polynomial ought to be intuitively connected to the edge weight, it clearly needs further refinement.

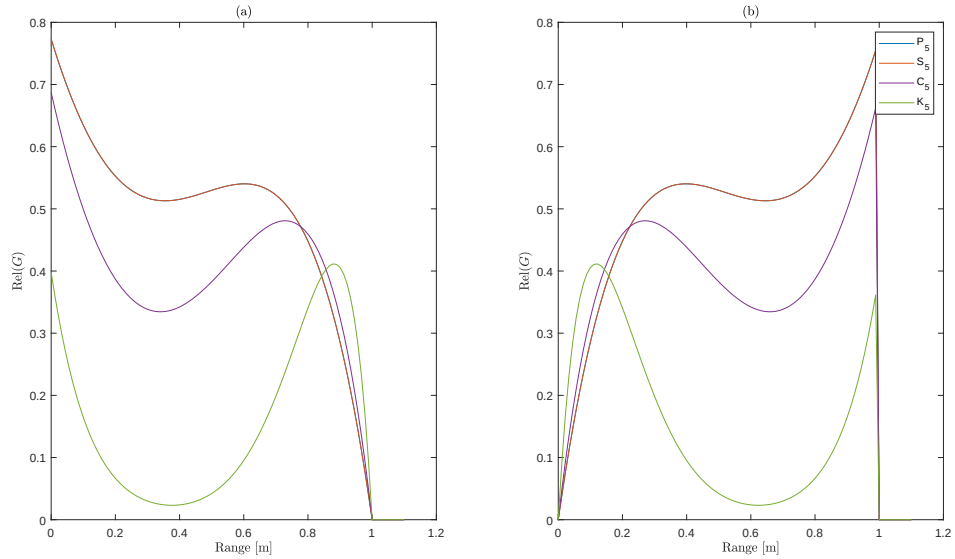

Figure 3.9. The reliability polynomial Rel(G) vs. range.

The *reliability polynomial* Rel(G) is shown relative to the range for a path graph, a star graph, a cycle graph, and a complete graph for (a) an increasing edge weight distribution (Eq. 3.1) and (b) a decreasing edge weight distibution (Eq. 3.5) that are linear functions of the range. The edge weight is used as the probability p in the computation of the reliability polynomial.

Increasing robustness is directly correlated to increasing the algebraic connectivity, which is reflected by the fact that the complete graph K_5 has the highest algebraic connectivity in Figure 3.10. A decreasing edge weight distribution is best used with the algebraic connectivity as shown in Figure 3.10(b).

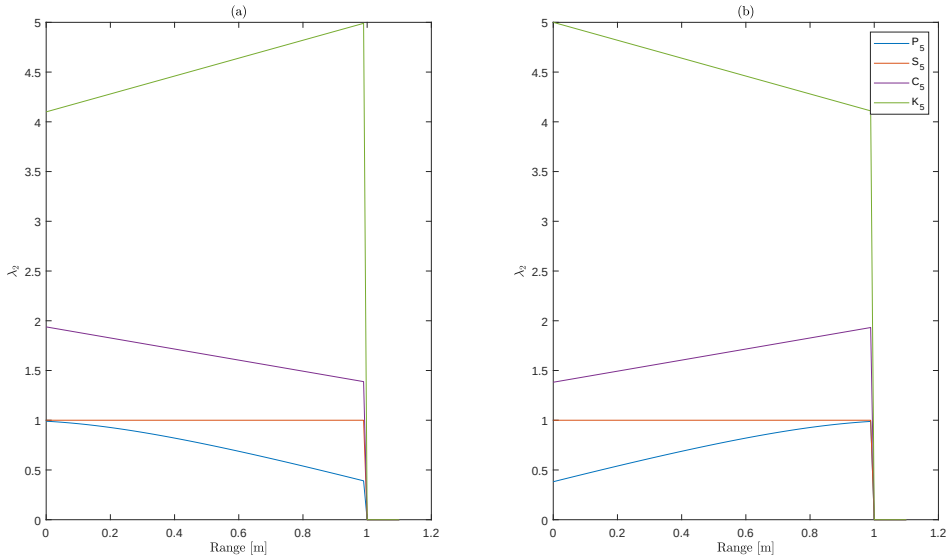

Figure 3.10. The algebraic connectivity $\lambda_2$vs. range.

The *algebraic connectivity* λ_2 is shown relative to the range for a path graph, a star graph, a cycle graph, and a complete graph for (a) an increasing edge weight distribution (Eq. 3.1) and (b) a decreasing edge weight distibution (Eq. 3.5) that are linear functions of the range.

Increasing the robustness of a graph reduces its effective resistance. Thus, as the distance between nodes becomes larger, we wish to see the effective resistance also increase. For this reason an increasing edge weight distribution is best used to quantify the robustness as shown in Figure 3.11(a). Note, when the graph becomes disconnected, the effective resitance goes to zero, thus this minima must be ignored in an optimization scheme.

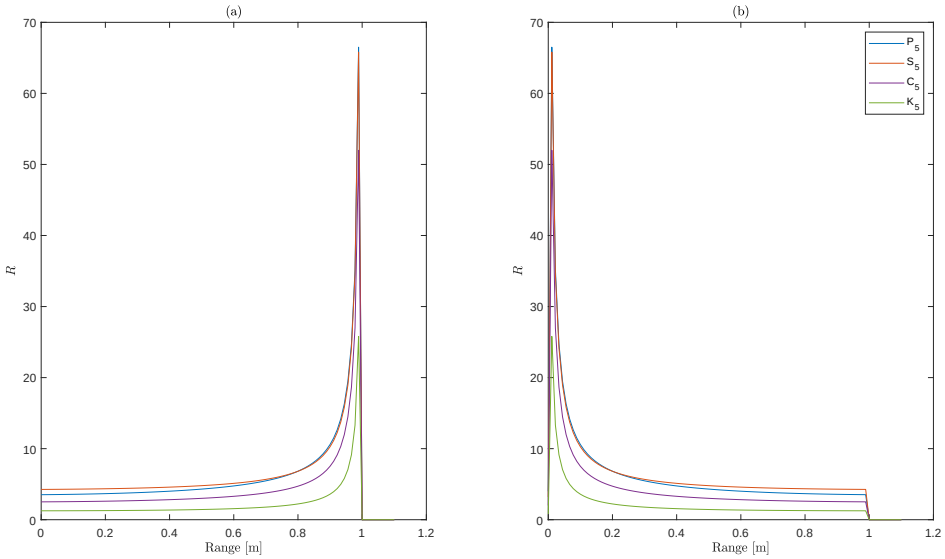

Figure 3.11. The effective resistance R vs. range.

The *effective resistance* R is shown relative to the range for a path graph, a star graph, a cycle graph, and a complete graph for (a) an increasing edge weight distribution (Eq. 3.1) and (b) a decreasing edge weight distibution (Eq. 3.5) that are linear functions of the range.

Overall, the effective graph resistance appears to be the best metric for quantifying the robustness of a weighted graph. It has good resolution to variations in the edge weight and is also strictly decreasing as edge weights are added. Additionally, it accounts for multiple paths between nodes (because of its relation to parallel resistors). For these reasons, we use it in the utility function described in Section 4.3.

Table 3.2. Distribution analysis for robustness metrics.

	Most Robust Value	Best Distribution Type	Multiple Paths?
$d_{\max}$	$\min(d_{\max})$	Increasing	No
$\bar{d}$	$\min(\bar{d})$	Increasing	No
E	$\max(E)$	Increasing	No
C	$\max(C)$	Decreasing	No
$\text{Rel}(G)$	$\max(\text{Rel}(G))$	Decreasing	-
λ_2	$\max(\lambda_2)$	Decreasing	No?
R	$\min(R)$	Increasing	Yes

The best edge weight distribution type (increasing or decreasing) is indicated that produces an intuitive change in the metric as a graph grows more or less robust. In general, we also desire a metric that is either reflects the added robustness from multiple paths. The results for the reliability polynomial are inconclusive. It is also unclear how the algebraic connectivity relates to quantifying the existence of multiple paths between nodes.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Problem Formulation

In this chapter we present the foundational mathematics for our simulation. We begin by describing time-varying formation control in Section 4.1. We then present submodularity and describe submodular function maximization in Section 4.2. We conclude by describing the utility function used to compute the optimal formation and prove its submodularity in Section 4.3.

4.1 Formation Control

We implement a position-based formation control to shape our network. In this formulation, individual agents are able to localize relative to a global reference frame. The agents actively control their position as prescribed by a centralized control algorithm that provides a specific location (and velocity) to actuate toward. One may note that interaction between agents is not required for the agents to move to a desired location; however, the control performance can be improved by including interactions between agents [17]. We also address issues of limited sensing and communications. In certain scenarios, agents may operate in a GPS denied environment. By using formation control, with at least one agent capable of localizing globally, all the agents can also localize globally by localizing relative to that one agent. We also decentralize control as much as possible. For a centralized coordination scheme, formation control is not required since it is not necessary to communicate if each agent is given instructions from the centralized controller. However, by using formation control, our system benefits by receiving feedback from neighboring agents and we lay the groundwork to decentralize control in future iterations.

We design our formation controller as a secondary controller that acts on top of the primary controllers on board the heterogeneous agents that comprise the NCS. Thus, we model each system with double-integrator particle dynamics [17]–[19]. Consequently, the only information that necessarily must be exchanged between agents is individual position and velocity information. We treat uncontrolled nodes (i.e., manned platforms and the NSW unit) as *virtual leaders*. The system does not generate a control input for these nodes, but the formation is recomputed in response to their movement.

Consider a mobile NCS consisting of N agents. We represent this NCS as a graph G. To simplify notation, we represent the graph Laplacian $\mathbf{L}(G)$ simply as $\mathbf{L}$. The off-diagonal elements of $\mathbf{L}$ (the edge weights w_{ij}) represent the interaction strength between agent i and agent j. We assume that G has a spanning tree, which implies that the system is stabilizable (see [18] for a proof). In our control we will also use the eigenvalues of $\mathbf{L}$, $\lambda_i, i \in 1, 2, \ldots, N$. Note if G has a spanning tree, $\lambda_1 = 0$.

We then approximate the dynamics of each agent $i = 1, 2, \ldots, N$ with a double integrator particle dynamics

$$\dot{\mathbf{r}}_i(t) = \mathbf{v}_i(t)$$

$$\dot{\mathbf{v}}_i(t) = \mathbf{b}_i\mathbf{u}_i(t),$$

where $\mathbf{r}_i(t) = [x_i(t),\ y_i(t)]^{\mathrm{T}}$ and $\mathbf{v}_i(t) = [u_i(t),\ v_i(t)]^{\mathrm{T}}$ represent the two-dimensional position and velocity vectors respectively, and $\mathbf{b}_i\mathbf{u}_i(t) \in \mathbb{R}^2$ relates the control input to the acceleration. We represent the individual agent dynamics in state-space as an LTI system

$$\dot{\mathbf{x}}_i(t) = \mathbf{A}\mathbf{x}_i(t) + \mathbf{B}\mathbf{u}_i(t) \tag{4.1}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \tag{4.2}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ b_{i,31} & 0 \\ 0 & b_{i,42} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

and $\mathbf{x}_i(t) = [\mathbf{r}_i(t),\ \mathbf{v}_i(t)]^{\mathrm{T}}$. In general, we assume the dynamics are consistent in the horizontal plane (i.e., $b_{i,31} = b_{i,42}$). In the special case of homogeneous agents, for any two agents i and j

$$b_{i,31} = b_{j,31},$$

and

$$b_{i,42} = b_{j,42}.$$

These equalities do not hold for heterogeneous systems where the dynamics are highly

disparate and cannot be approximated as equivalent particles (e.g., an UAV compared to an UUV). For homogeneous systems, we can concisely write the dynamics of the entire NCS,

$$\dot{\mathbf{x}} = \mathbf{I}_N \otimes \mathbf{A}\mathbf{x} + \mathbf{I}_N \otimes \mathbf{B}\mathbf{u}, \tag{4.3}$$

where $\otimes$ indicates the Kronecker product, $\mathbf{x}(t) = [\mathbf{x}_1(t),\ \mathbf{x}_2(t),\ \ldots,\ \mathbf{x}_N(t)]^{\mathrm{T}}$, and $\mathbf{u}(t) = [\mathbf{u}_1(t),\ \mathbf{u}_2(t),\ \ldots,\ \mathbf{u}_N(t)]^{\mathrm{T}}$.

We define the time-varying formation as $\mathbf{h}(t) = [\mathbf{h}_1(t),\ \mathbf{h}_2(t),\ \ldots, \mathbf{h}_N(t)]^{\mathrm{T}}$. Here $\mathbf{h}_i(t)$ is the piecewise continuously differentiable vector, $\mathbf{h}_i(t) = [\mathbf{h}_{i,\mathbf{r}},\ \mathbf{h}_{i,\mathbf{v}}]^{\mathrm{T}}$ which indicates the position and velocity assigned to agent i in the formation. New formation configurations $\hat{\mathbf{h}}(t)$ are computed at discrete intervals separated by some Δt as described in Section 4.2. The continuous function $\mathbf{h}(t)$ is then defined as a linear interpolation between the configurations $\hat{\mathbf{h}}(t)$ and $\hat{\mathbf{h}}(t + \Delta t)$.

From here we will simplify our notation by indicating $\mathbf{x}_i(t)$ with $\mathbf{x}_i$, $\mathbf{u}_i(t)$ with $\mathbf{u}_i$, $\mathbf{h}_i(t)$ with $\mathbf{h}_i$ etc. For the the system defined by Equation 4.1, Dong et al. propose the control law

$$\mathbf{u}_i = \mathbf{K}_1[\mathbf{x}_i - \mathbf{h}_i] + \mathbf{K}_2 \sum_{j=1}^{N} \left[w_{ij} \left(\mathbf{x}_j - \mathbf{h}_j \right) - (\mathbf{x}_i - \mathbf{h}_i) \right] + \dot{\mathbf{h}}_{i,\mathbf{v}}. \tag{4.4}$$

In this control law, the feedback gains $\mathbf{K}_1$ and $\mathbf{K}_2$ are used to shape the response of the time-varying formation center and the response of individual agents respectively. For homogeneous nodes, the complete system can then be defined as

$$\dot{\mathbf{x}} = [\mathbf{I}_N \otimes (\mathbf{A} + \mathbf{B}\mathbf{K}_1) - \mathbf{L} \otimes (\mathbf{B}\mathbf{K}_2)]\mathbf{x} - [\mathbf{I}_N \otimes (\mathbf{B}\mathbf{K}_1) - \mathbf{L} \otimes (\mathbf{B}\mathbf{K}_2)]\mathbf{h} - [\mathbf{I}_N \otimes \mathbf{B}]\dot{\mathbf{h}}_{\mathbf{v}}. \tag{4.5}$$

A necessary and sufficient condition for the NCS defined by Equation 4.5 to achieve the time-varying formation $\mathbf{h}$ is provided in [19] along with a proof.

Theorem 1 *The system 4.5 achieves time-varying formation if and only if for any* $i \in \{1, 2, \ldots, N\}$

$$\lim_{t \to \infty} \left[(\boldsymbol{h}_{i,\boldsymbol{v}} - \boldsymbol{h}_{j,\boldsymbol{v}}) - (\dot{\boldsymbol{h}}_{i,\boldsymbol{v}} - \dot{\boldsymbol{h}}_{j,\boldsymbol{v}}) \right] > 0, j \in N_i \tag{4.6}$$

and for any $i \in \{2, 3, \ldots, N\}$

$$-k_{12} + Re(\lambda_i)k_{22} > 0 \tag{4.7}$$

$$(-k_{12} + Re(\lambda_i)k_{22})\psi_i - Im(\lambda_i)^2 k_{21}^2 > 0 \tag{4.8}$$

where

$$\psi_i = k_{12}k_{11} - Re(\lambda_i)(k_{12}k_{21} + k_{11}k_{22}) + (Re(\lambda_i)^2 + Im(\lambda_i)^2)k_{21}k_{22}.$$

Dong et al. then provide a procedure to select the gain matrices in Equation 4.4. First, using a pole-placement technique, select desired locations in the open left-hand plane for the poles of $\mathbf{A} + \mathbf{B}\mathbf{K}_1$ to determine $\mathbf{K}_1$ [26]. This controls the dynamics of the formation center. Then, using Theorem 2, compute $\mathbf{K}_2$, which satisfies the conditions in Theorem 1, as proven in [19].

Theorem 2 *If condition 4.6 in Theorem 1 holds, then system 4.5 achieves time-varying formation with the control law 4.4 where*

$$K_2 = \frac{1}{Re(\lambda_2)}\boldsymbol{B}^T\bar{\boldsymbol{P}},$$

where $\bar{\boldsymbol{P}}$ *is the positive definite solution to the algebraic Riccati equation*

$$\bar{\boldsymbol{P}}(\boldsymbol{B}\boldsymbol{K}_1 + A) + (\boldsymbol{B}\boldsymbol{K}_1 + \boldsymbol{A})^T\bar{\boldsymbol{P}} - \bar{\boldsymbol{P}}\boldsymbol{B}\boldsymbol{B}^T\bar{\boldsymbol{P}} + \boldsymbol{I} = 0$$

4.2 Submodular Function Maximization

In this discussion we refer to the work of Krause and Golovin in [24] and [1], and of Summers, Cortesi, and Lygeros in [2] for further details on the properties of set functions, discussions of submodularity, and examples of maximization and adaptive submodularity.

4.2.1 Submodularity

A *set function* $f : 2^V \rightarrow \mathbb{R}$ maps a subset $S \subseteq V$ to a value $f(s)$ [24]. In our problem, the *ground set* V is the discretized set of locations where we can assign nodes. The subset S is then the set of N locations where we place our N nodes (the agents in our NCS). The set function $f(S)$ computes the utility of placing the nodes at locations S.

Submodularity is a property of set functions commonly referred to as the property of diminishing returns; that is adding a node to subset of B, will produce a larger utility gain than adding the node to B [24]. Krause and Golovin provide two definitions of submodularity as provided in Definition 4.2.1. The first of these two definitions relies on the *discrete derivative* which we define before defining submodularity. In addition, we define a subclass of submodular functions, *monotone* functions.

Definition 4.2.1 *Discrete Derivative: For a set function $f : 2^V \to \mathbb{R}$, $S \subseteq V$, and $v \in V$, let $\Delta_f(v|S) := f(S \cup v) - f(S)$ be the* discrete derivative *of f at S with respect to v [24].*

If it is clear that the function f is implied, we simplify notation by writing the discrete derivative without the subscript as $\Delta(v|S)$.

Definition 4.2.2 *Submodularity: A set function $f : 2^V \to \mathbb{R}$ is* submodular *if for every $A \subseteq B \subseteq V$ and $v \in V \setminus B$ it holds that*

$$\Delta_f(v|A) \geq \Delta_f(v|B). \tag{4.9}$$

Equivalently, a function $f : 2^V \to \mathbb{R}$ is submodular *if for every $A, B \subseteq V$ [24],*

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B). \tag{4.10}$$

Definition 4.2.3 *Monotonicity: A function $f : 2^V \to \mathbb{R}$ is* monotone *if for every $A \subseteq B \subseteq V$, $f(A) \leq f(B)$ [24].*

4.2.2 Maximization

We are interested in finding the set S of node locations that will maximize the utility function $f(S)$. That is we wish to solve

$$\max_{S \subseteq V} f(S). \tag{4.11}$$

on which we will place constraints [24]. For instance, we are limited by the number of agents N available. This is referred to as a *cardinality constraint*, i.e., $|S| \leq N$. This

problem is Non-deterministic Polynomial-time (NP) hard [24]. To demonstrate this fact, consider our problem of maximizing the utility of placing N nodes. We discretize our area of interest into an r by c two-dimensional grid containing p locations ($p = r \times c$). In order to find the set of locations maximizing the utility function $f(s)$, we must consider p^N permutations. Clearly, this quantity grows exponentially as more nodes are added and the grid space is discretized into a finer mesh.

Fortunately, this problem, subject to cardinality constraints, can be approximately solved utilizing a *greedy algorithm*. Starting with the empty set S_0, the greedy algorithm adds the element e that maximizes the discrete derivative $\Delta_f(v|S_{i-1})$ at each iteration $i = 0, 1, \ldots, N$

$$S_i = S_{i-1} \cup \{\arg\max_v \Delta_f(v|S_{i-1})\}. \tag{4.12}$$

This significant result was proven by Nemhauser et al. in 1978 [32]. An extension to this proof is provided in [24] to allow for cases where the greedy algorithm places more nodes K than the optimal number of nodes N.

Theorem 3 (Nemhauser et al. 1978 [32]) *Fix a nonnegative monotone submodular function $f : 2^V \to \mathbb{R}_+$ and let $\{S_i\}_{i\geq 0}$ be the greedily-selected sets defined in Equation 4.12. Then for all positive integers N and K,*

$$f(S^*) \geq \left(1 - e^{-K/N}\right) \max_{S:|S|\leq N} f(S). \tag{4.13}$$

In particular, for $K = N$, $f(S_N) \geq (1 - 1/e) \max_{|S|\leq k} f(S)$.

From this we derive two important conclusions. First, if we design our utility function to be a nonnegative monotone submodular function, we are guaranteed to asymptotically approach the optimal solution using a greedy algorithm that can be executed in polynomial time. Second, as the number of nodes increases, the closer our approximation approaches the optimal solution. Golovin and Krause note that if the utility of a certain number of nodes N are determined to be a sufficient optimal utility $f(S^*)$, the greedy algorithm continues to asymptotically approach the optimal value if it is allowed to place more than N nodes. We display this property in Figure 4.1, where the blue line indicates the lower bound of the greedy algorithm as a fraction of the optimal utility $f(S^*)$.

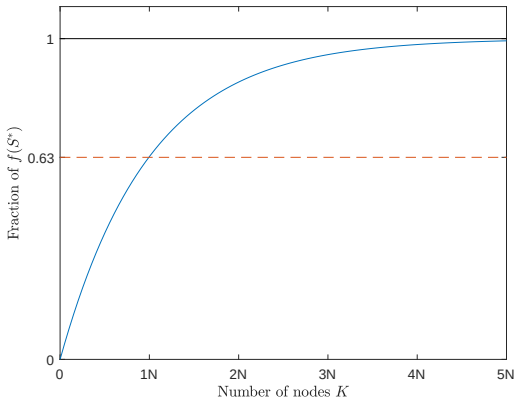

Figure 4.1. Asymptotic property of the greedy algorithm.

As the number of nodes K is increased, the lower bound of the utility produced by the greedy algorithm approaches the optimal $f(S^*)$. If exactly N nodes are placed, the utility will be no less than 63% of the optimal (indicated in red). If $5N$ nodes are placed, the utility improves to 99%. The lower bound of the greedy algorithm's performance is indicated in blue.

By utilizing the greedy algorithm, we reduce the complexity of our problem from evaluating $f(S)$ for p^N permutations, to instead evaluating $f(S)$ only $p \times N$ times. Krause and Golovin point out that in some cases, the evaluation of $f(S)$ is still computationally intensive and render even the greedy algorithm infeasible. For this they propose and explore the use of lazy evaluations to form an accelerated greedy algorithm [24].

4.3 Utility Function

We purposely design a monotone submodular utility function in order to reduce the computational complexity of our optimization problem and take advantage of the bounded performance of the greedy algorithm. One useful property we will use in the design of our utility function is that submodularity is preserved for linear combinations of submodular functions. To clarify, if set functions $g_1, g_2, \ldots, g_n : 2^V \rightarrow \mathbb{R}$ are submodular and $\alpha_1, \alpha_2, \ldots, \alpha_n \geq 0$ are nonnegative coefficients, then $f(S) = \sum_{i=1}^{n} \alpha_i g_i(S)$ is submodular [24]. We use this property to design a complex utility function that is composed of a linear combination of simpler utility subfunctions. Our utility function J is composed of

components which quantify the sensing ability $f_s(S)$ and the communication robustness of the network $f_r(S)$:

$$J(S) = \alpha_s f_s(S) + \alpha_r f_r(S). \tag{4.14}$$

In addition, mobility and dynamics constraints are considered in the utility function. We enforce these constraints by setting the values of the utility function to zero for locations that would place a node in violation of its physical constraints. For each subfunction, submodularity is preserved as long as the utility values are greater than zero, thus submodularity is preserved while enforcing these constraints [24]. In future iterations, we may consider gramian-based controllability and observability utility subfunctions proposed in [2].

One point of consideration is where to place the first node. In all the cases we examine, virtual leaders are already present in the environment. Consequently, the node is typically placed within communication range of a virtual leader (depending on the values of α_s and α_r). However, if no virtual leaders were present, the first node would simply be placed in the location that provides the greatest sensing utility. Then subsequent nodes are placed within communication range of that node. If there is no absolute maxima, then the greedy algorithm arbitrarily selects one of the maxima and continues placing nodes.

4.3.1 Sensing Subfunction

We are interested in the ability of our network to sense the environment and provide coverage of areas of interest. That is, we wish to select a subset of $V = 1, 2, \ldots, p$ locations to place N nodes. If we place node i at location j, we say that it provides a sensing benefit of $\Theta_{i,j}$, where $\mathbf{\Theta} \in \mathbb{R}^{N \times p}$. If each node is assigned to the location with the largest benefit, the total value is the set function

$$f_s(S) = \sum_{i=1}^{N} \max_{j \in S} \Theta_{i,j} \tag{4.15}$$

If $\Theta_{i,j} \geq 0$ for all i, j, then $f_s(S)$ is monotone submodular [24]. In our scenario we determine the value of the $\Theta_{i,j}$ based on the characteristics of node i and the proximity of location j to points of interest. In general, we use a similar function to Equation 3.3. In this case,

however, the range r is the distance to the nearest point of interest and $r_{i,\max}$ is the maximum sensing range of node i

$$\Theta_{i,j}(r) = \begin{cases} \frac{w_j}{2}\left(1 + \cos\left(\pi \frac{r - r_{i,\text{suf}}}{r_{i,\max} - r_{i,\text{suf}}}\right)\right) & \text{if } r \leq r_{i,\text{suf}} \\ 0 & \text{if } r > r_{\max} \end{cases}. \tag{4.16}$$

where w_j represents the relative importance of sensing the point of interest at location j and $r_{i,\text{suf}}$ quantifies the distance at which the sensing utility is not improved by approaching closer to the point of interest.

For example, we assume that UAV nodes can effectively travel anywhere in the discretized space and we assign a relative importance between monitoring the road network and specific targets using the weight w_j. In order to compute $\boldsymbol{\Theta}$ we use image processing techniques to extract the roads from satellite imagery and use a distance transform to determine the distance to the nearest section of road [33]. We then add targets for the UAV to investigate on the northwest and southeast ends of the island (indicated with red asterisks in Figure 4.2). We can represent this sensing function as a heat map, as demonstrated in Figure 4.2(c).

The rows of $\boldsymbol{\Theta}$ that relate to the sea-based USV and UUV are computed similarly, but with one key difference. These nodes clearly cannot drive ashore, and commanding them to do so would lead to the loss of an asset. Consequently, all values are automatically set to zero so that the submodular maximization function will not place a USV or UUV node on land. Further safeguards are of course incorporated in those agents' primary controllers to prevent them from entering shoal water. The development and graphical representation of the row in $\boldsymbol{\Theta}$ relating to an UUV is shown in Figure 4.3. Figure 4.3 demonstrates that we are primarily interested in searching the coast for hazards and extraction routes. We can bias the node towards searching a particular location that we believe to be of particular interest, as demonstrated in the South East corner of the island in Figure 4.3.

By formulating the sensing utility subfunction in this manner, we accomplish two objectives. First, we bias our network toward points of interest that we can identify explicitly, or we can allow the network to identify these automatically based on feature recognition software. Second, we design this such that nodes are not commanded to go where they cannot physically go. These demonstrations provide a snapshot of the sensing subfunction.

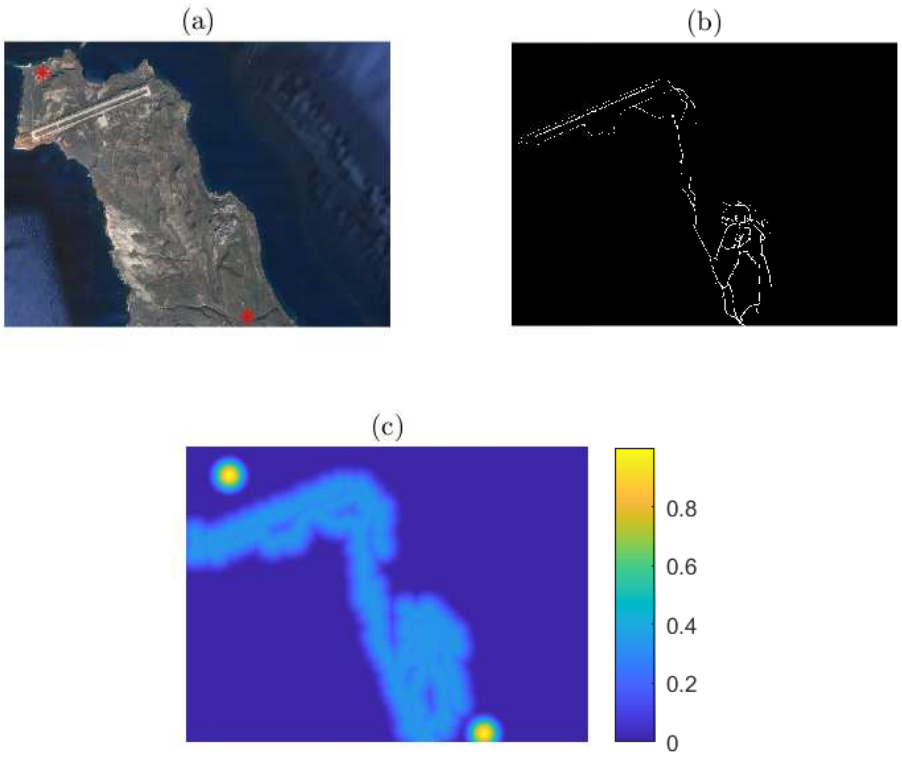

Figure 4.2. UAV sensing utility subfunction.

The sensing utility $\Theta_{i,j}$ assigned to UAV i for all locations $j = 1, 2, \ldots, p$. (a) Satellite imagery is analyzed to extract (b) points of interest such as roads, and (c) an utility is assigned to each discretized point in accordance with Equation 4.16.

However, the matrix $\boldsymbol{\Theta}$ in reality is time-dependent. As the nodes move through the environment, the data they collect can inform $\boldsymbol{\Theta}$ to either increase or decrease the interest in location j if something needing further investigation is surveyed or if the area has been marked clear of adversaries. We account for this fact by lowering the values of $\Theta_{k,j}$ for nodes $k = i + 1, i + 2, \ldots, n$ after node i has been placed in the locations j near to node i. This ensures that the network provides coverage of all regions of interest. In future iterations, this sensing model could be made significantly more complex. For instance, as the nodes progress through the environment, recently covered areas would have a low value; but as time goes by and the certainty of measurements made at location j decrease, the attractiveness would increase.

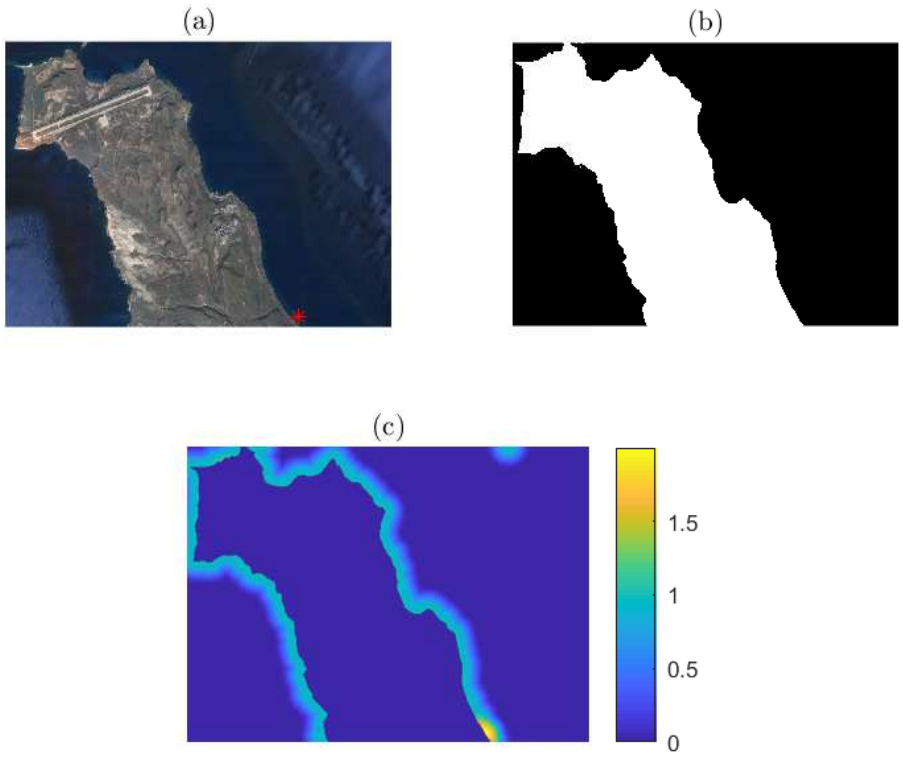

Figure 4.3. UUV sensing utility subfunction.

The sensing utility $\Theta_{i,j}$ assigned to UUV i for all locations $j = 1, 2, \ldots, p$. (a) Satellite imagery is analyzed to (b) eliminate the land and (c) an utility is assigned to each discretized point to highlight the coastline and other areas of interest in accordance with Equation 4.16.

4.3.2 Robustness Subfunction

We use the effective graph resistance (described in Chapter 3, to quantify the robustness of the graph. We use the inverse exponential edge weight distribution as described by Equation 3.6. By using an inverse exponential edge weight, the effective graph resistance increases roughly linearly as the average distance between nodes increases.

Since the graph becomes more robust as the effective resistance decreases, we manipulate

the resitance to form a new metric $\mathbf{\Omega}$ that conforms to a maximization problem

$$f_r(S) = \sum_{i=1}^{N} \max_{j \in S} \Omega_{i,j} \tag{4.17}$$

where $\Omega_{i,j} = 1 - \text{norm}(R_{i,j})$. Here, $R_{i,j}$ is the effective resistance if a node i is added to the network at location j. We then normalize the resistance from zero to one such that $0 \leq \Omega_{i,j} \leq 1$. When $\Omega_{i,j}$ is at a maximum ($\Omega_{i,j} = 1$), graph resitance is at a minimum, meaning the network is at its most robust. Since $\Omega_{i,j} \geq 0$ for all i, j, this function is monotone submodular [24].

By using this metric and a communications-based edge weight, we attempt not only to maximize the robustness of our network, but we also incorporate optimizing communications into the placement of our nodes. We can further improve this relationship by using communication strength (e.g., SNR) to compute the edge weights instead of the range which only roughly approximates the signal strength.

4.3.3 Mobility constraints

In some iterations we also consider the mobility of our nodes. Although our submodular maximization function executes in near real-time, it operates at a much slower rate than the measurement and control systems on board each agent. For this reason, it is used to generate new formation $\mathbf{h}_r(t)$ every Δt. This formation is used as part of a secondary controller that operates on top of the agents' primary controllers.

We wish this new formation to be reachable within Δt. For this reason, we limit how far the maximization function can move node i in one time step. In general, we limit this displacement based on the node's maximum velocity to $\max(\mathbf{v}_i)\Delta t$. For a location outside of this maximum displacement, we set $\Theta_{i,j}$ and $\Omega_{i,j}$ to zero, such that the function will never direct a node beyond its reachable sphere of influence.

We examine the effects of varying Δt. We wish to compare the formations generated for varying update rates and for cases where this constraint is dropped. We also examine the affects of placing more restricted nodes first (i.e., nodes with a slower maximum velocity) or last (Section 5.3).

4.4 Summary

In this chapter, we have introduced the mathematics used to control our NCS. A greedy algorithm is used to generate new formations at discrete intervals. This greedy algorithm selects each formation to maximize a utility function that encapsulates the networks ability to sense its operating environment and communicate robustly. The agents that comprise the NCS are controlled using a time-varying formation controller.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Results and Analysis

This chapter focuses on the results and analysis of the submodular maximization function for a mobile NCS. The goal is to provide near-optimal time varying formation solutions that position nodes to support competing objectives.

We first demonstrate the performance of our submodular maximization function for a single time step (i.e., without formation control) in Sections 5.1, 5.2, 5.3, and 5.4. In Section 5.1, we demonstrate our utility function is tuned and show how varying the utility subfunction weights induces topological changes. Then in Section 5.2, we demonstrate how the greedy algorithm iteratively selects node locations. We also examine how altering the order of placing heterogeneous nodes affects the formation generated in Section 5.3. Finally, we enforce reachability constraints when placing nodes in Section 5.4.

In Section 5.5, we compare the solution of the submodular maximization function to the optimal solution computed using an exhaustive brute force approach. We conclude by demonstrating the coupled performance of the formation controller and maximization function as the virtual leaders move through operational phases (Section 5.6).

5.1 Utility Function Tuning

Overall, when evaluating the greedy algorithm and our submodular utility function, we are looking for the formation it generates to appear *logical* above all. We refer to Chapter 16 of [34] for a thorough discussion of this complex evaluative measure; however, for our purposes the generated formation should provide coverage over areas of interest and maintain connectivity between all nodes. This is where the the weights on sensing and robustness utility, α_s and α_r, become useful because we use them to tune the generate formation to produce a rational and desirable solution that balances the competing objectives during each phase of operation (See Section 5.6).

Before we can allow this NCS to operate autonomously, we must first build trust in the system. Even if a formation maximizes the utility function, if it is irrational, trust will not

be warranted. Thus it is of utmost importance that the control emulates (and exceeds) the performance of a human decision maker. Comparatively, it is easy for the algorithm to place a substantial number of nodes faster than a human; however, the NCS must also provide sufficient utility to the operators it is supporting. As initially envisioned the NCS would act in the background and simply provide recommendations to the operators of the unmanned agents. As confidence in the recommendations of the NCS grows, it can be given more autonomy to control the physical systems.

We began tuning the utility function with a simple network composed of homogeneous UAV nodes. This allows us to simplify communications and sensing models such that they were equivalent for all of the nodes in the network. In Figure 5.1(a) we show the near-optimal placement of five UAVs relative to two virtual leaders (the NSW unit and DDG) and sensing targets. Each node is placed greedily in sequence to maximize the utility function. To get a better intuition of the shape of the utility function, we plot the total utility that led to the placement of the most recent node for all discrete locations j as a heat map in Figure 5.1(b). This total utility is the weighted sum of the sensing utility in Figure 5.1(c) and the robustness utility in Figure 5.1(d). A feature of note for the sensing utility is observable in Figure 5.1(c); once a node is placed, the utility for all the locations in its "field of view" are set to zero.

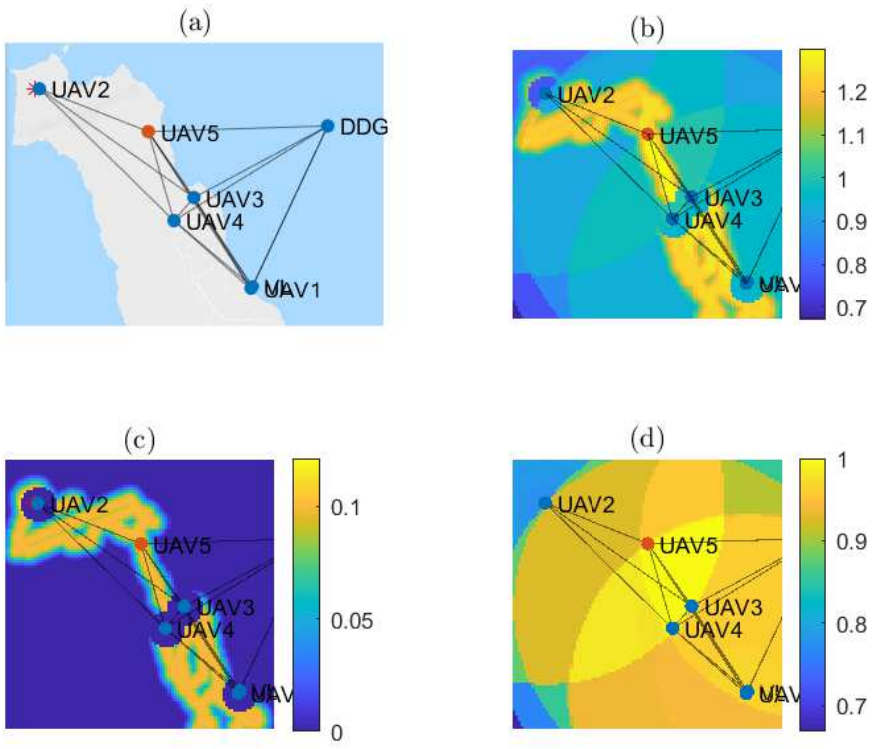

Figure 5.1. Formation with homogeneous agents.

(a) The network is shown with the most recently-placed node highlighted in red. This node was placed at the point maximizing the utility function. (b) The total utility shown as a heat map. (c) A heat map of the sensing utility $\Theta_{i,j}$ shown for all locations j. (d) A heat map of the robustness utility $\Omega_{i,j}$ shown for all locations j.

After validating our optimization scheme with homogeneous agents, we expanded the scheme to address heterogeneous agents. We demonstrate the placement of heterogeneous nodes in Section 5.2. While tuning the utility function, it became evident that the utility subfunction weights α_s and α_r could be employed to induce significant topological changes in the network. This fact is demonstrated in Figure 5.2.

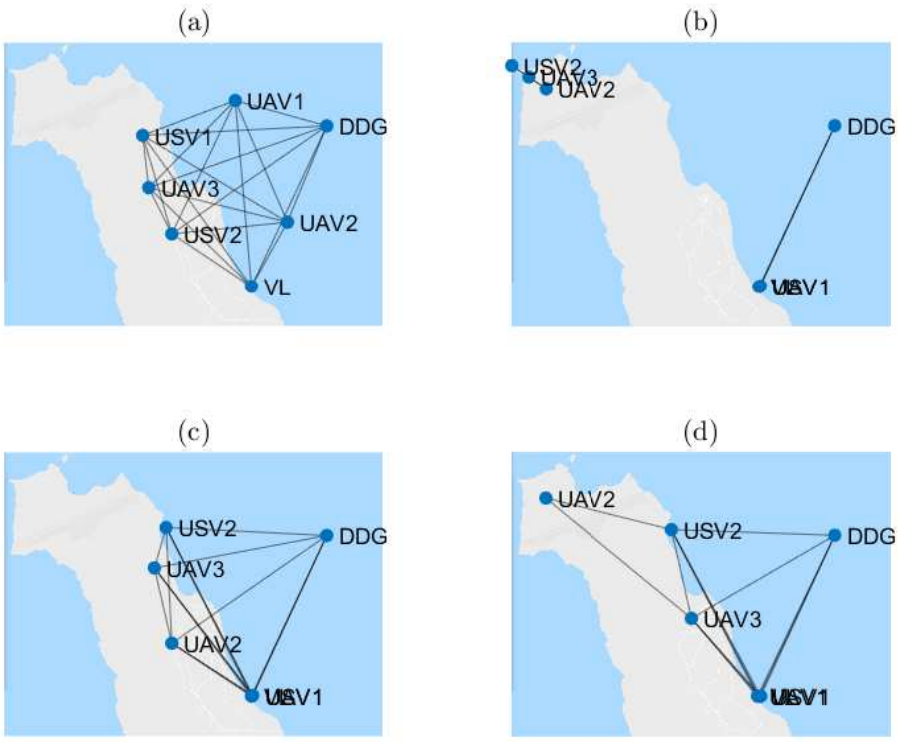

Figure 5.2. Comparison of varying utility subfunction weights.

Topological changes in the network are shown. These changes are induced by varying the utility subfunction weights. In (a) $\alpha_s = 0$ and $\alpha_r = 1$. In (b) $\alpha_s = 1$ and $\alpha_r 0$. In (c) $\alpha_s = 1$ and $\alpha_r = 1$. In (d) $\alpha_s = 3$ and $\alpha_r = 1$.

In Figure 5.2(a), all of the emphasis is placed on maximizing robust communications with respect to the virtual leaders (i.e., $\alpha_s = 1$ and $\alpha_r 0$). This causes the agents to cluster in a formation that maximizes the number of communications links. Figure 5.2(b) demonstrates the polar opposite case, where $\alpha_s = 1$ and $\alpha_r 0$. Here, the nodes are placed to simply maximize the each agent's ability to sense the environment. When equal emphasis is placed on each subfunction ($\alpha_s = 1$ and $\alpha_r = 1$), the formation shown in Figure 5.2(c) results.

We deemed that weighting each utility subfunction equally did not provide sufficient coverage of the island. Thus, in Figure 5.2(d) we show the case where $\alpha_s = 3$ and $\alpha_r = 1$. We consider this to be a potentially optimal balance of the competing objective to sense the

environment and communicate robustly. In Chapter 6, we describe adaptive submodularity, which can be used to select optimal *policies*. An example of a policy, could be this specific set of subfunction weights. Rather than tuning the parameters based our own observations, adaptive submodularity could be used to adjust these weights based on a probabilistic model of the world state and measurements.

5.2 Iterative Node Placement

We now considered heterogeneous agents. This requires us to enforce the mobility constraints of each node. Whereas for UAV nodes we assume their airspace is unrestricted, for sea-based nodes we ensure that they are not directed to ground themselves by zeroing the sensing utility above land. Even for an extremely low-weighted sensing utility as in Figure 5.3(c), this prevents nodes from being directed aground. We also enforce a standoff distance from the shore by using a dilation when we use image processing to parse out land from the ocean [33]. In Figure 5.4 we continue placing aerial assets after the sea-based assets shown in the previous figure. This creates a well-connected network that maintains communications as well as overwatch on the target and the NSW unit.

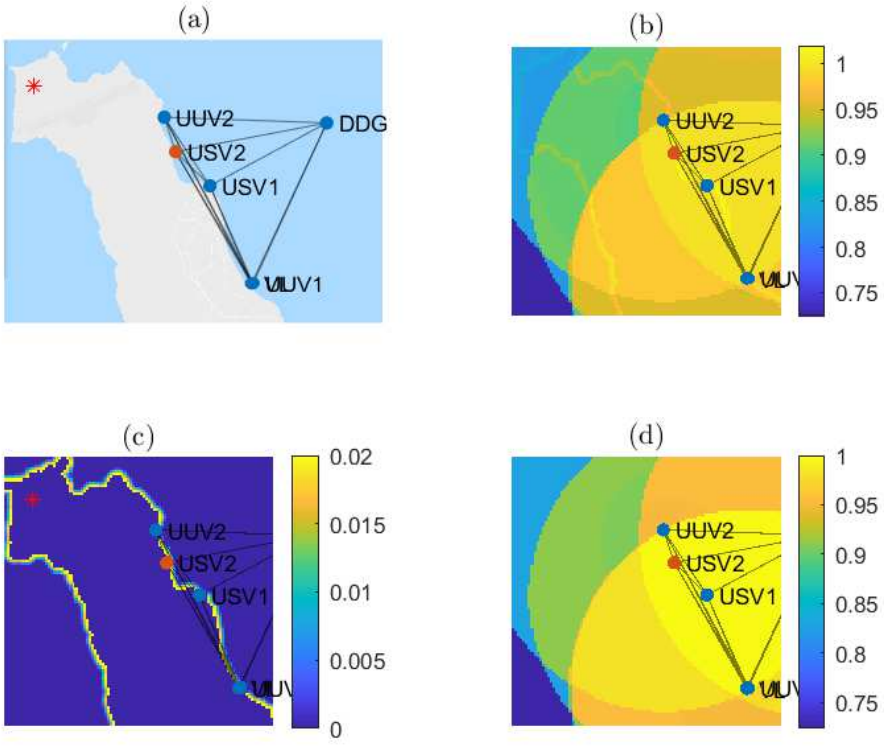

Figure 5.3. Formation with half heterogeneous agents placed.

(a) The network is shown with the most recently-placed node highlighted in red. This node was placed at the point maximizing the utility function. (b) The total utility shown as a heat map. (c) A heat map of the sensing utility $\Theta_{i,j}$ shown for all locations j. (d) A heat map of the robustness utility $\Omega_{i,j}$ shown for all locations j.

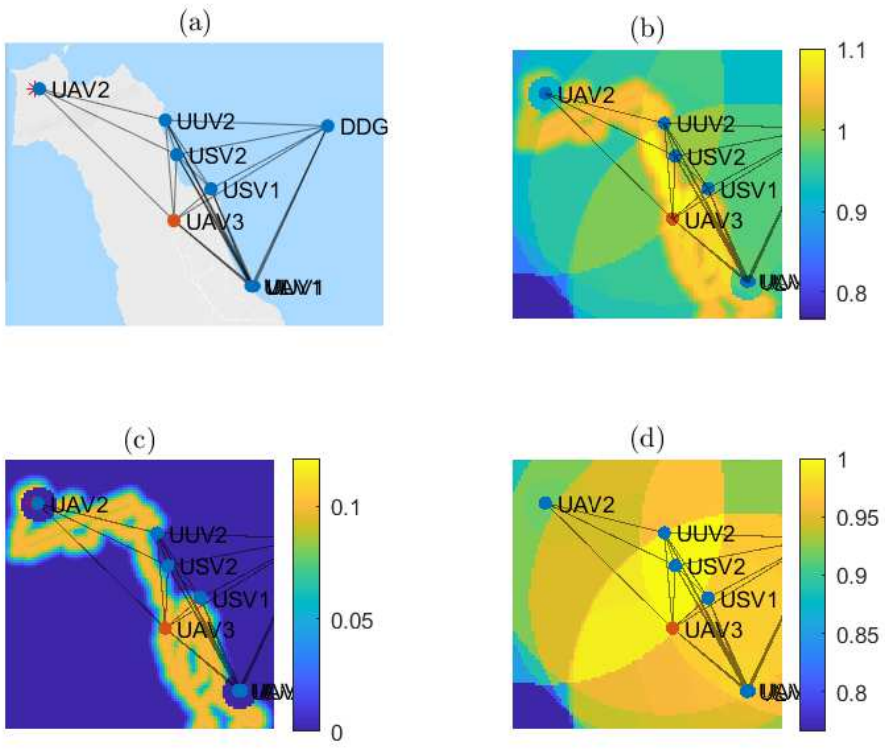

Figure 5.4. Formation with all heterogeneous agents placed.

(a) The network is shown with the most recently-placed node highlighted in red. This node was placed at the point maximizing the utility function. (b) The total utility shown as a heat map. (c) A heat map of the sensing utility $\Theta_{i,j}$ shown for all locations j. (d) A heat map of the robustness utility $\Omega_{i,j}$ shown for all locations j.

5.3 Node Placement Order

For the previous heterogeneous networks, we placed nodes according to the somewhat-arbitrary rule of placing the most constrained (either slowest or having the smallest operating zone) first. We hypothesized that this would result in better formations and that the order of placing nodes would significantly affect the network topology. In Figure 5.5, we reversed the placement order from Figures 5.3 and 5.4. Instead we place the most mobile UAV

nodes first. Although some specific nodes trade places, the final network formation created is surprisingly similar.

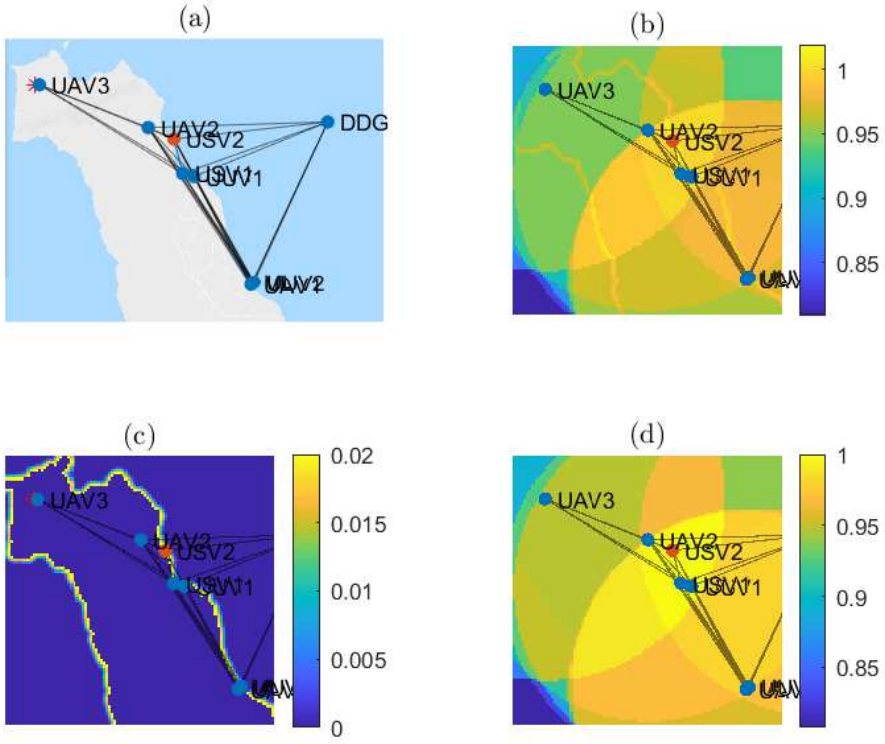

Figure 5.5. Heterogeneous formation with reversed placement order.

(a) The network is shown with the most recently-placed node highlighted in red. This node was placed at the point maximizing the utility function. (b) The total utility shown as a heat map. (c) A heat map of the sensing utility $\Theta_{i,j}$ shown for all locations j. (d) A heat map of the robustness utility $\Omega_{i,j}$ shown for all locations j.

5.4 Reachability Constraints

We further improve our formation generator by limiting its output to include only locations in each node's reachable workspace, i.e., a new formation is not useful if none of the nodes can reach that formation by the time the next new formation is generated. This is enforced by limiting the output of the utility function to only include locations that are within a

certain range of the node's current location. This is demonstrated in Figure 5.6(b) where the majority of the map is ruled out as an option when reassigning the position for UAV2. The algorithm thus must select whatever local maxima is located within a sufficient range of the node. This range is computed based on the maximum speed of the node and the time step that the algorithm is prognosticating forward. In Figure 5.7 we show the network's current position, and the newly-commanded formation based on a 100 second prognostication.

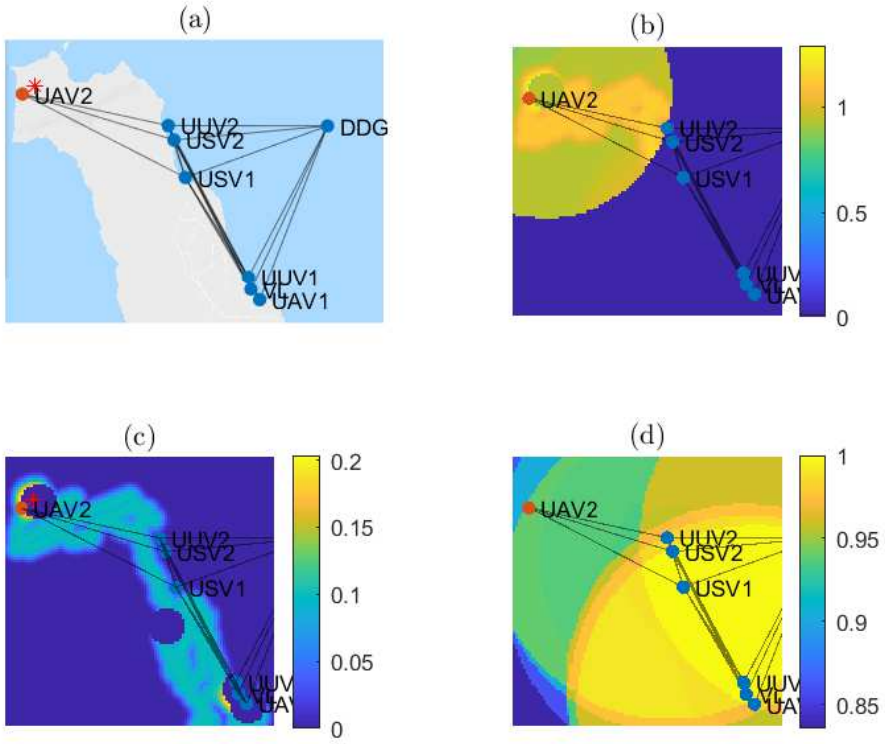

Figure 5.6. Formation with placement constrained by current position.

(a) The network is shown with the most recently-placed node highlighted in red. This node was placed at the point maximizing the utility function. (b) The total utility shown as a heat map, where only reachable locations are allowed. (c) A heat map of the sensing utility $\Theta_{i,j}$ shown for all locations j. (d) A heat map of the robustness utility $\Omega_{i,j}$ shown for all locations j.

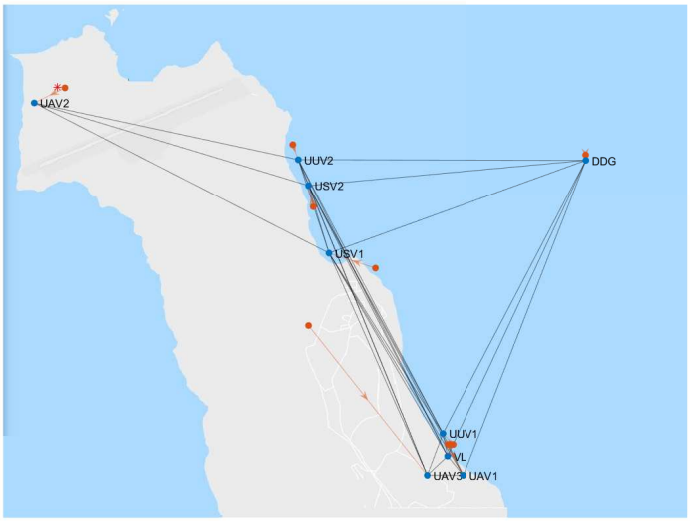

Figure 5.7. Heterogeneous formation with current position shown.

The current formation and subsequent formation is shown for a 100 second prognostication. The current node locations are indicated in red, with arrows pointing to the newly assigned position.

Overall, we determine that our utility function meets our criteria of rationality in that it consistently generates a formation that meets the objectives to maximize sensing and communications.

5.5 Comparison to Brute Force Approach

We now present a simple comparison of our maximization function to the brute force approach. Due to the sheer complexity of computing the absolute maximum value of the utility function, we were limited to finding the maximum utility for the placement of five nodes in a grid with only 130 grid points (in all other cases we used a grid with 1000 points). A brute force algorithm was designed to exhaustively compute all possible configurations of the network. The optimal formation computed using the brute force algorithm required over 3.7×10^{10} computations of the utility function. This formation is shown in Figure 5.8 with the grid overlaid on the map.

Figure 5.8. Optimal formation generated with brute force.

The optimal network configuration is shown for a simple grid containing 130 discretized points. It required 3.7×10^10 evaluations of the utility function to determine the optimal configuration with utility $J(S^*) = 0.9895$.

In comparison, the greedy algorithm only required 650 utility function evaluations and produced a very similar formation. However, the most telling result is the fact that $J(S^*) = 0.9895$, while the greedy algorithm found a formation with a nearly identical utility $J(S) = 0.9820$. This formation and grid is overlaid on the map of SCI in Figure 5.9. Overall, we consider this a remarkable success and proof of concept.

Figure 5.9. Near-optimal formation generated with the greedy algorithm.

The near-optimal network configuration is shown for a small grid containing 130 discretized points. This only required 650 evaluations of the utility function to determine the optimal configuration with utility $J(S) = 0.9820$.

The advantages of submodular function maximization are even more obvious when we consider scaling up the number of agents or the number of discretized points. In reality a network consisting of five agents is very trivial, but it is extremely encouraging that the greedy algorithm performs so well. While it would be interesting to compare larger numbers agents and discretized points to the absolute maximum, the problem is entirely intractable as these numbers are increased.

5.6 Formation Control

In each phase of operation, the virtual leaders move independently of the network based on predefined tracks or their own volition. The other nodes in the network are controlled using the time-varying formation controller described by Equation 4.5. The formation $\mathbf{h}$ is adaptively recomputed based on the movement of the virtual leaders. The slave nodes do not have knowledge of the virtual leaders' intended path; only the virtual leader's current position when they are within communication range.

At the MTX, six phases of operation were conducted; IPB, Insertion, Infiltration, Actions at the Objective, Exfiltration, and Extraction. These phases require significant topological changes in the network. We induce these transformations in the network by tuning the parameters α_s and α_r based on the emphasis of sensing versus communications and robustness. The submodular function maximization is then used to recompute the formation. We constrain the utility function based on the heterogeneous properties of each node such that the nodes are not given an unreachable position in the formation. Initially, sensing is the most important factor and the network biases itself toward surveying the island. Gradually, the weight on the importance of communications and network robustness is increased, bringing the network closer together in support of the NSW unit.

During *IPB*, the network is composed solely of two UUVs, one USV, and one UAV. The unmanned assets conduct ISR and operate independently for the majority of the phase. The UUVs act as the virtual leaders during this phase, following pre-planned routes to survey the seafloor where the NSW unit plans to infiltrate the island. At the beginning of the phase, there is no need to communicate ($\alpha_r = 0$) and the nodes act separately to survey the environment, as shown in Figure 5.10(a). As more information is gathered, we gradually

increase α_r and the nodes draw closer together in Figure 5.10(b) (Note: this is very close based on the assumed communication ranges of the UUVs).

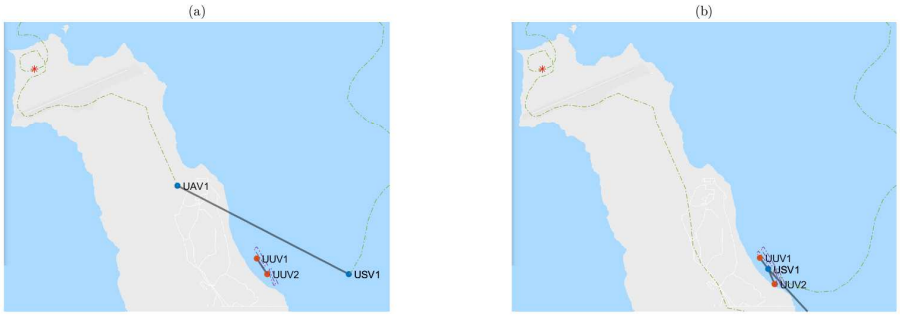

Figure 5.10. Network during initial IPB.

(a) The network is shown during beginning of the IPB phase. (b) The network is shown at the end of the IPB phase. USV1 has traveled close enough to transmit information gathered by the UUVs. Communication links form and break as the nodes leave and enter communication range. The virtual leaders are indicated in red and their paths are indicated in purple. The formation controller directs the other nodes shown in blue, with their paths indicated in green. Communication links are shown in gray.

Following IPB, the NSW unit and support ship (DDG) enter the network, along with other unmanned assets. During *Insertion*, the NSW unit uses another USV to travel to shore, which also acts as the virtual leader. The support ship also acts independently, so the network treats it as an additional virtual leader. The movements of the network are shown in Figure 5.11. Throughout the phase, robustness is weighted more heavily and the nodes are moved to provide better connectivity. One may note the UUV nodes are not pictured because they have left the network.

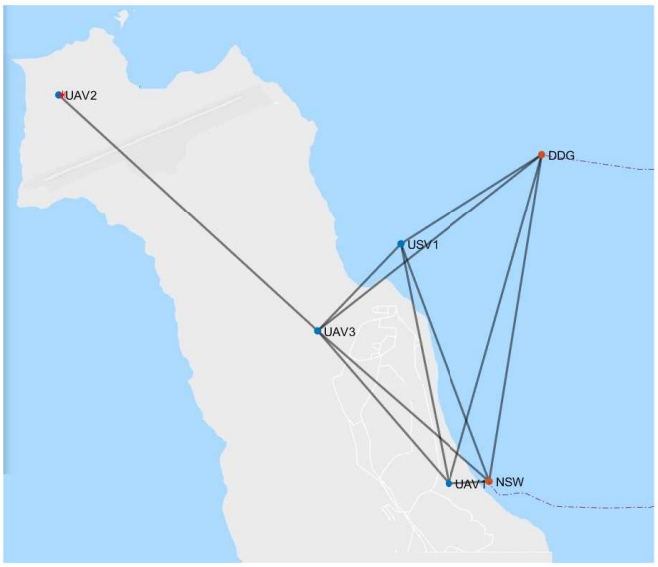

Figure 5.11. Network during the Insertion phase.

The network is shown near the end of the NSW Insertion. The virtual leaders are indicated in red and their paths are indicated in purple. The formation controller directs the other nodes shown in blue. Communication links are shown in gray.

After the NSW unit has been inserted, the unit conducts *Infiltration* and move on foot to conduct *Actions at the Objective*. The infiltration is the longest phase of the operation, and we show multiple panes separated by approximately 20 minutes of real time in Figure 5.12 to demonstrate the topological changes of the network. In the series of snapshots, the NSW unit is shown traversing toward the target indicated by the red star on the northwest end of the island. Meanwhile, the DDG circles offshore. For the entire operation, the network adapts its formation to the movement of the two virtual leaders and maintains sensing coverage and robust communications. As time goes on, the importance of robustness is increased, bringing the network in tighter around the NSW unit, rather than providing extensive sensing coverage of the island.

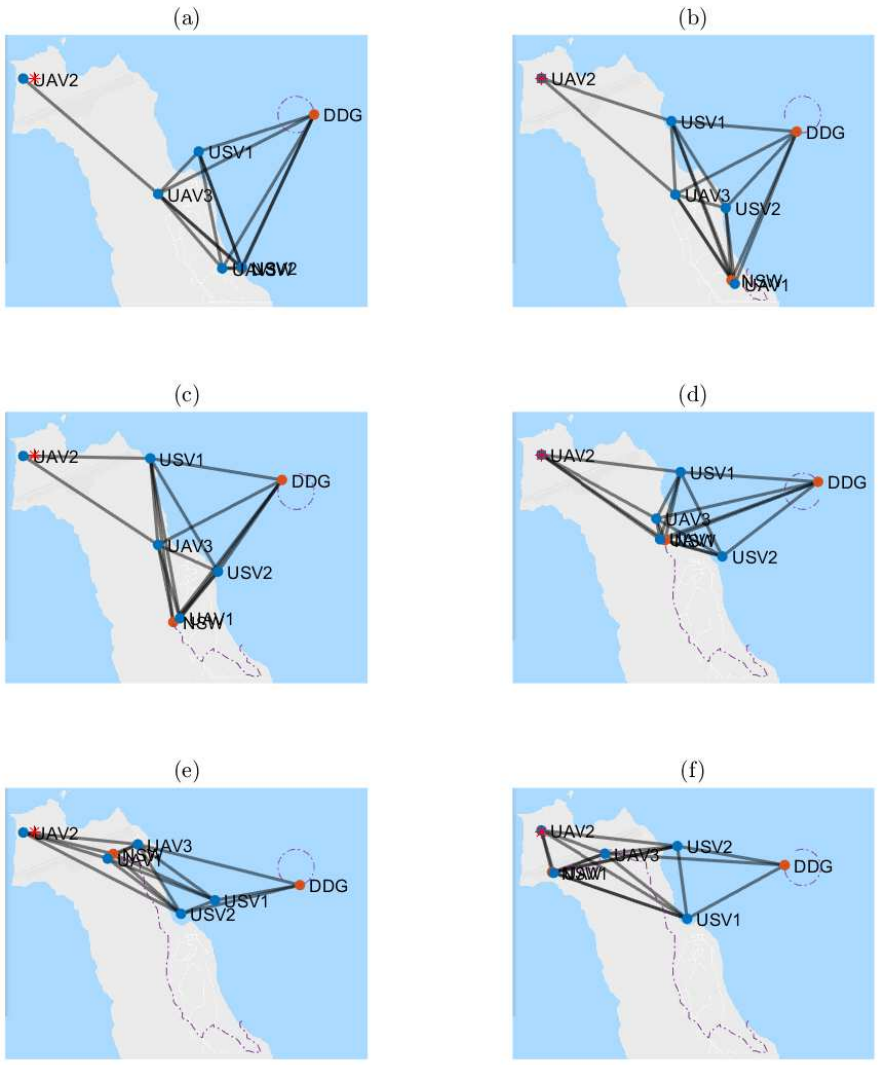

Figure 5.12. Network during the Infiltration phase.

The network throughout the Infiltration. In the progression from (a) to (f), the NSW unit traverses up the island and the network responds to its movement and the movement of the DDG as it circles offshore. The virtual leaders are indicated in red and their paths are indicated in purple. The formation controller directs the other nodes shown in blue. Communication links are shown in gray.

Once the NSW unit has infilled to the target, the unit conducts *Actions at the Objective* and then reverse course during *Exfiltration* and is *Extracted*. The network is not demonstrated during Actions at the Objective due to the phase's brevity. The Exfiltration and Extraction phases are not pictured due to their similarity to the previous phases, conducted in reverse. Although we frame this in terms of the MTX, these results can be generalized to many other scenarios. This demonstrates the feasibility of using a time-varying formation controller with heterogeneous nodes. The controller also handles the addition and loss of nodes as they leave and enter the network, seamlessly breaking and reestablishing communications.

Additionally, this demonstrates the ability of submodular function maximization to balance competing objectives and efficiently solve a complex combinatorial optimization problem. In this simulation, the map is discretized into a 100 by 100 grid. At most time steps there were, at minimum, five mobile nodes. Using a brute force method, 1×10^{20} permutations would need to be considered. This is completely infeasible with the hardware onboard any of the agents in the network, and extremely time intensive even when using a supercomputer. Instead, this problem is reduced to a scalable problem that is solvable in polynomial time.

CHAPTER 6:
Adaptive Submodularity

In this chapter we provide an overview of an adaptive submodular approach to use machine learning to improve the performance of our NCS. In Section 6.1, we describe adaptive submodularity, a technique that uses machine learning to adaptively adjust a policy to achieve an optimal outcome. We then synthesize our problem in terms of adaptive submodularity in Section 6.2. The implementation of this approach is left to future work.

6.1 Framework

In our problem, we may wish to use sensor readings or the actual received communication strength as feedback to *adaptively* place the next node. In this formulation, rather than merely optimizing over a utility function, we optimize over a *policy*. A policy is a function that translates from gathered information to the next action taken [1].

The following is an enumeration of the variables relating to adaptive submodularity and their definitions as defined in [24] and [1].

Actions V: There is a finite set of actions V (e.g., node location selections) available to us. We call a single action v and a set of specific actions $A \subseteq V$.

Outcomes O: A finite set of outcomes O correspond to the finite set of available actions V. For example, the outcome of a certain NCS topology could be the resultant band width of the network.

World State Φ: The random variable Φ is used to model the state of the world. For instance, this could represent the likelihood that each location j is mined.

Realization ϕ: A realization ϕ is a concrete state from the probabilistic world state Φ, e.g., there are mines at locations i, j, and k. We use the world state ϕ as a function that maps from an action to an outcome, i.e., $\phi : V \rightarrow O$. Thus, $\phi(v)$ is the outcome of action v. For example, if the world is in state ϕ and we take an action by placing a node at location v, we realize the outcome $\phi(v)$ (e.g., the realization that there is or is not a mine at v).

Prior Probability Distribution $\mathbb{P}(\phi)$: Using a Bayesian model, we assume that there is a prior probability distribution $p(\phi) = \mathbb{P}(\Phi = \phi)$. We use this model to predict *a posteriori* realizations, i.e., the results of the set of actions A. This distribution is akin to the sensing utility described in Section 4.3; that is this distribution helps us generate prediction of the expected gain in utility.

Partial Realization ψ: We define a partial function ψ that maps the actions performed to the observed outcomes. We use $\text{dom}(\psi)$ to denote the domain of ψ, i.e., the set of actions A performed up to the current time. The *partial realization* $\psi(A)$ is the set of outcomes observed up to the current time (e.g., the set of measurements from locations A).

Objective Function f: We design an objective function $f : 2^V \times O^V \rightarrow \mathbb{R}_+$ that computes the utility $f(A, \phi)$ of the actions A resulting in the particular realization of the world state ϕ.

Policy π: A policy π is a function that maps from the partial realizations ψ to new actions. Thus, $\pi(\psi)$ are the actions taken by the policy π after observing the partial realization ψ. We denote the set of actions played by policy π under realization ϕ as $V(\pi, \phi)$. An example of a policy could be a certain set of gains $\alpha_1, \ldots, \alpha_n$ that are used to tune the utility function to produce a different network topology.

Expected utility $f_{\text{avg}}(\pi)$: The expected utility of policy π is

$$f_{\text{avg}}(\pi) = \mathbb{E}\left[(V(\pi, \Phi), \Phi)\right] = \sum_{\phi} \mathbb{P}[\phi]\ f(V(\pi, \phi), \phi). \tag{6.1}$$

Now that we have setup each of the relevant variables, we describe the objective of *Adaptive Stochastic Maximization*. We wish to maximize the utility of a limited number of actions; that is we wish to find a policy π^*

$$\pi^* \in \arg\max_{pi} f_{\text{avg}}(\pi), \tag{6.2}$$

subject to cardinality constraints, i.e., $|V(\pi, \phi)| \leq k$ for all ϕ [24]. Krause and Golovin show that for this problem of Adaptive Stochastic Maximization, if the utility function f is adaptive monotonic and adaptive submodular, bounded performance of the greedy algorithm generalizes to the adaptive situation [24]. The adaptive greedy algorithm functions similarly

to the standard greedy algorithm and relies on the *conditional expected marginal benefit*, which is similar to the discrete derivative [1].

Definition 6.1.1 *Conditional Expected Marginal Benefit: given a partial realization ψ and an action v, the* conditional expected marginal benefit *of v is*

$$\Delta(v|\psi) = \mathbb{E}\left[f(dom(\psi) \cup \{v\}, \Phi) - f(dom(\psi), \Phi)|\psi \subseteq \Phi\right] \tag{6.3}$$

and the conditional expected marginal benefit *of a policy π is*

$$\Delta(\pi|\psi) = \mathbb{E}\left[f(dom(\psi) \cup V(\pi, \Phi), \Phi) - f(dom(\psi), \Phi)|\psi \subseteq \Phi\right]. \tag{6.4}$$

While less than k actions have been taken, the *adaptive greedy algorithm* determines the action v^* that maximizes the conditional expected marginal benefit. Action v^* is then taken. The realization $\phi(v^*)$ is observed and then added to the partial realization ϕ [24]. Adaptive submodularity has the advantage in that it can be used to deal with uncertainty, i.e., when the world state realization is not deterministic.

6.2 Application

In our problem, we are limited by the resources available to us. We are limited by the number of unmanned agents, the amount of fuel on each agent, the sensing range, the communications range, and the time available to execute a mission. We also force the constraint that we have limited knowledge of our operating environment. For these reasons, adaptive submodularity is a natural extension to the approach we have discussed in previous chapters. We can deploy our mobile sensing platforms individually, one by one. Then using adaptive submodularity, we can utilize the intelligence collected by each deployed agent to inform the placement of each successive node. In this scheme we can attempt to preserve our resources and maximize the reduction of our uncertainty about the world state.

In this problem (as before), we discretize our area of interest into a set of locations V. We aim to select the set of locations A that maximizes the certainty in our representation of the environment. However, before we deploy our agents the state of the environment is unknown

or at most partially known. Thus, we assign an initial state $\phi(v)$ for each discretized point v. This initial state could indicate the probability that there is an adversary at that location.

We then design an objective function $f : 2^V \times O^V \rightarrow \mathbb{R}_+$ that quantifies the utility of positioning nodes at locations A given the realization ϕ of those points. Using f, we aim to adaptively position our limited number of k nodes in order to maximize the intelligence we gather about the environment. We can then use a Bayesian approach to estimate the full world state from our partial realizations.

CHAPTER 7:
Conclusion

In this concluding chapter we summarize our findings (Section 7.1) and provide an overview of future extensions to this thesis (Section 7.2).

7.1 Contributions

In this thesis we address a broad range of topics. Significantly, we develop novel criteria for the evaluation of robustness metrics. We then develop new network robustness metrics based on weighted graphs. We conclude that the effective graph resistance metric best encapsulates network robustness in a single value.

We also develop a utility function that quantifies competing sensing and communication objectives. We prove that this function is submodular and use it compute optimal network formations with respect to a realistic operation. We then demonstrate that the greedy algorithm indeed produces a near optimal solution by comparing its output to that of the brute force computation of the full power set.

Time-varying formation control is then demonstrated in simulation. In this simulation, we show that the agents are able to adaptively reconfigure in response to the movements of virtual leaders.

We then formulate our problem in terms of adaptive submodularity, a promising approach that could be used to tune the performance of the control algorithm to produce more desirable results. Although we leave the implementation of this to future work, this holds the potential to make our controller scalable to networks containing significantly more node where tuning the utility subfunction gains becomes more difficult.

Potential applications of this research are far ranging. The framework developed in this thesis could be applied to any number of scenarios in which manned and unmanned agents work to achieve certain objectives that can be formulated as a coverage problem. Such operations include humanitarian operations, search and rescue, or the spreading of pesticides to manage an invasive species or disease vector such as mosquitoes.

7.2 Future Work

Overall, with future work we aim to make our control algorithm more robust to failure and uncertainty. In Chapter 6 we outline our approach to implement adaptive submodularity to address the uncertainty in the environment. In order to make our network more resistant to failure, we seek to decentralize control and reduce the need for communication.

As currently implemented, the formation of our nodes is computed in a centralized fashion. Complete positional knowledge and connection information is required to calculate the utility of our network. This could be improved by using estimators to track node positions in the case of brief network dropouts and using a more realistic communications model to ensure commanded formations will not cause nodes to become disconnected. Protocols also need to be designed for scenarios in which nodes become disconnected from the network.

In addition, the utility function could possibly be improved by using Gramian-based controllability and observability metrics proposed in [2]. Other factors that were not considered, but would be useful additions to the utility function, include the energy usage and time required to reconfigure the formation. The capacity of the network for deception could also be considered. For instance, using noncritical nodes to lead adversaries away from more valuable nodes, or ensuring that nodes acting as overwatch for ground forces do not give away the position of those assets.

This research will also be translated from simulation into physical hardware. Using the Robotic Operating System (ROS) as the middleware for the NCS, the described control architecture will be implemented on the assets described in Chapter 1 and evaluated at future MTX events [35].

List of References

[1] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *arXiv*, pp. 1–60, Dec. 2017.

[2] T. H. Summers, F. L. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 91–101, Mar. 2016.

[3] W. Ellens and R. Kooij, "Graph measures and network robustness," *arXiv*, pp. 1–12, Nov. 2013.

[4] X.-M. Zhang, Q.-L. Han, and X. Yu, "Survey on recent advances in networked control systems," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1740–1752, Oct. 2015.

[5] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.

[6] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, July 1987.

[7] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[8] H. Li, J. Peng1, W. Liu1, J. Wang, J. Liu1, and Z. Huang, "Flocking control for multi-agent systems with communication optimization," presented at IEEE American Control Conference, Washington, DC, 2013.

[9] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of IEEE*, vol. 95, no. 1, pp. 215–233, Mar. 2007.

[10] R. M. Murray, "Recent research in cooperative control of multivehicle systems," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, May 2007.

[11] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[12] S. Knorn, Z. Chen, and R. H. Middleton, "Overview: Collective control of multiagent systems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 4, pp. 334–347, Aug. 2015.

[13] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 1–7, Nov. 2017.

[14] J. Qin, Q. Ma, Y. Shi, and L. Wang, "Recent advances in consensus of multi-agent systems: A brief survey," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4972–4983, June 2017.

[15] X. Ge, Q.-L. Han, D. Ding, X.-M. Zhang, and B. Ning, "A survey on recent advances in distributed sampled-data cooperative control of multi-agent systems," *Neurocomputing*, vol. 275, pp. 1684–1701, Jan. 2018.

[16] M. Schuresko and J. Cortés, "Distributed motion constraints for algebraic connectivity of robotic networks," presented at IEEE Conference on Decision and Control, Cancun, Mexico, 2009.

[17] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, no. 53, pp. 424–440, Mar. 2015.

[18] G. Lefferriere, A. Williams, J. S. Cuaghman, and J. Veerman, "Decentralized control of vehicle formations," *Mathematics and Statistics Faculty Publications and Presentations*, no. 142, pp. 243–255, 2004.

[19] X. Dong, B. Yu, Z. Shi, and Y. Zhong, "Time-varying formation control for unmanned aerial vehicles: Theories and applications," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 340–348, Jan. 2015.

[20] J. Cortés, "Coverage optimization and spatial load balancing by robotic sensor networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 749–754, Feb. 2010.

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

[22] K. Laventall and J. Cortés, "Coverage control by multi-robot networks with limited-range anisotropic sensory," presented at IEEE American Control Conference, Seattle, WA, 2008.

[23] P. Agharkar and F. Bullo, "Quickest detection over robotic roadmaps," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 252–259, Feb. 2016.

[24] A. Krause and D. Golovin, "Submodular function maximization," in *Tractability: Practical Approaches to Hard Problems*, L. Bordeaux, Y. Hamadi, and P. Kohli, Eds. Cambridge: Cambridge University Press, 2014, pp. 71–104.

[25] Y.-Y. Liu and A.-L. Barabási, "Control principles of complex systems," *Reviews of Modern Physics*, vol. 88, pp. 1–55, Sep. 2016.

[26] N. S. Nise, *Control Systems Engineering*, 7th ed. California State Polytechnic University, Pomona: Wiley, 2015.

[27] Y.-Y. Liu1, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, p. 167–173, May 2011.

[28] X. Yang1, Y. Zhu1, J. Hong, L.-X. Yang1, Y. Wu1, and Y. Y. Tang, "The rationality of four metrics of network robustness: A viewpoint of robust growth of generalized meshes," *PLoS ONE*, pp. 1–13, Aug. 2016.

[29] A.-L. Barabási, *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. New York: Plume, 2003.

[30] A. Esfahanian, "Connectivity algorithms," course notes for CSE 835 Algorithmic Graph Theory, Deptartment of Computer Science and Engineering, Michigan State University, Monterey, CA, spring 2017.

[31] D. G. Harris and F. Sullivan, "Linear algebra and sequential importance sampling for network reliability," National Institute of Standards and Technology, Tech. Rep., Dec. 2011.

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

[33] V. Curic, "Mathematical morphology and distance transforms," Centre for Image Anlysis, Swedish University of Agricultural Sciences, 2010.

[34] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, NY: Pearson Edication, Inc., 2003.

[35] "Robotic operating system," June 2018. [Online]. http://www.ros.org.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California