# Large-Scale Dynamic Observation Planning for Unmanned Surface Vessels

by

John V. Miller

B.S. Operations Research
United States Air Force Academy, 2005

SUBMITTED TO THE SLOAN SCHOOL OF MANAGEMENT IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN OPERATIONS RESEARCH
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2007



Signature of Author: _____
Sloan School of Management
Interdepartmental Program in Operations Research
May 17th, 2007

Approved by: _____
David W. Carter
The Charles Stark Draper Laboratory, Inc.
Technical Advisor

Approved by: _____
Stephan E. Kolitz
The Charles Stark Draper Laboratory, Inc.
Technical Supervisor

Certified by: _____
Professor James B. Orlin
Edward Pennell Brooks Professor of Operations Research
Thesis Advisor

Accepted by: _____
Professor Cynthia Barnhart
Professor of Civil and Environmental Engineering Department
Engineering Systems Division
Co-director, Operations Research Center

**[This Page Intentionally Left Blank]**

# Large-Scale Dynamic Observation Planning for Unmanned Surface Vessels

by

John Vaala Miller

Submitted to the Sloan School of Management on
May 17th, 2007 in partial fulfillment of the requirements for the
Degree of Master of Science in Operations Research

## Abstract

With recent advances in research and technology, autonomous surface vessel capabilities have steadily increased. These autonomous surface vessel technologies enable missions and tasks to be performed without the direction of human operators, and have changed the way scientists and engineers approach problems. Because these robotic devices can work without manned guidance, they can execute missions that are too difficult, dangerous, expensive, or tedious for human operators to attempt. The United States government is currently expanding the use of autonomous surface vessel technologies through the United States Navy's Spartan Scout unmanned surface vessel (USV) and NASA's Ocean-Atmosphere Sensor Integration System (OASIS) USV. These USVs are well-suited to complete monotonous, dangerous, and time-consuming missions. The USVs provide better performance, lower cost, and reduced risk to human life than manned systems.

In this thesis, we explore how to plan multiple USV observation schedules for two significant notional observation scenarios, collecting water temperatures ahead of the path of a hurricane, and collecting fluorometer readings to observe and track a harmful algal bloom. A control system must be in place that coordinates a fleet of USVs to targets in an efficient manner.

We develop three algorithms to solve the unmanned surface vehicle observation-planning problem. A greedy construction heuristic runs fastest, but produces suboptimal plans; a 3-phase algorithm which combines a greedy construction heuristic with an improvement phase and an insertion phase, requires more execution time, but generates significantly better plans; an optimal mixed integer programming algorithm produces optimal plans, but can only solve small problem instances.

Technical Supervisor: Stephan E. Kolitz
The Charles Stark Draper Laboratory, Inc.

Thesis Advisor: Professor James B. Orlin
Edward Pennel Brooks Professor of Operations Research

**[This Page Intentionally Left Blank]**

# ACKNOWLEDGMENTS

I would like to thank all of those individuals who have helped me throughout my life, including my family, friends, teachers, and mentors. I would also like to thank the many individuals that I owe for my two years at Draper Laboratory and MIT:

First, I thank Dr. Stephan Kolitz of Draper Laboratory for his invaluable role in the completion of this thesis. I owe him many thanks for his dedication to my research. His expertise and insight were such a valuable source of guidance. I also thank him for finding advisors interested in my research. Without the extra time he was willing to commit to my research, my thesis research would not have been the same.

Second, I thank Dr. David Carter of Draper Laboratory for his patience and understanding. I owe him many thanks for his dedication to helping me edit my thesis and adding his technical expertise to the project.

I am also thankful for Charlie Strauss of Draper Laboratory. I am truly grateful for him helping me with Java programming.

I also want to thank Professor Jim Orlin for his help and expertise. I am greatly appreciative of his willingness to donate his limited time to my thesis and research.

I would also like to thank the Education Office at Draper Laboratory, the Operations Research Center at MIT, and the United States Air Force for the opportunity to continue my education at Draper Laboratory and MIT. In particular, I am grateful for Lt. Col. Andrew Armacost and Colonel Steve Baker at the United States Air Force Academy for their efforts in helping me secure this assignment.

To all of my friends that I met here in Boston that made my two years here enjoyable, I also say thank you. I give special thanks to Kiel, Joe, Kevin, Stephan, Keith, Chris, Jeff (for the extra monitor that saved me), and the other Air Force lieutenants. Without these friendships, I would not have made it through many of the challenges and opportunities over the past two years.

Finally, yet most importantly, I would like to thank my lovely wife Katie (whom I cherish with my whole heart), my parents, and the rest of my family. They have provided me with constant love and support. Through their faith, guidance, encouragement, and help, I have been able to accomplish so much.

The thesis was prepared at The Charles Stark Draper Laboratory, Inc. under Contract NAS5-03121 sponsored by the National Aeronautics and Space Administration's (NASA) Earth Science Technology Office's (ESTO) Advanced Information Systems Technology-02 (AIST) program and NASA's ESTO's AIST-05 program, under Grant Number NNX06AG17G.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or The U.S. Government.

John V. Miller, 2nd Lt., USAF May 17, 2007

**[This Page Intentionally Left Blank]**

# TABLE OF CONTENTS

**[This Page Intentionally Left Blank]**

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

# Introduction

With recent advances in research and technology, autonomous surface vessel capabilities have steadily increased. These autonomous surface vessel technologies enable missions and tasks to be performed without the direction of human operators, and have changed the way scientists and engineers approach problems. Because these robotic devices can work without manned guidance, they can execute missions that are too difficult, dangerous, expensive, or tedious for human operators to attempt. The United States government is currently expanding the use of autonomous surface vessel technologies through the United States Navy's Spartan Scout unmanned surface vessel (USV) and NASA's Ocean-Atmosphere Sensor Integration System (OASIS) USV. These USVs are well-suited to complete monotonous, dangerous, and time-consuming missions. The USVs provide better performance, lower cost, and reduced risk to human life than manned systems.

In this thesis, we explore how to plan multiple USV observation schedules for two significant notional observation scenarios, collecting water temperatures ahead of the path of a

hurricane, and collecting fluorometer readings to observe and track a harmful algal bloom. To properly manage and fully realize the capabilities of the USVs, a control system must be in place that directs a fleet of USVs to targets and coordinates the fleet in an efficient manner. The control system should be able to plan for many USVs over many thousands of targets. It also should create plans very quickly to allow for dynamic re-planning to react to changes in the environment and observation needs.

## 1.1 Thesis Overview

The goal of this research is to develop an automated observation planner for a fleet of USVs that allocates the fleet efficiently and takes advantage of the USV's true potential. We now present a chapter overview of the remainder of the thesis:

**Chapter 2 – Observing Dynamic Oceanic Phenomena Operational Description.** In this chapter, we present the problem of autonomous coordinated observation of dynamic oceanographic phenomena such as hurricanes and algae blooms. We describe ocean phenomena in general, two significant example observation scenarios, and how each phenomenon is currently observed. In addition, we describe the physical components of a notional sensor web that will enable enhanced observation methods. Furthermore, we sketch how the sensor web could be applied in each of the ocean observation scenarios. Finally, we describe the specific problem of unmanned surface vessel fleet path and observation planning that this thesis addresses.

**Chapter 3 – Model Development.** In this chapter, we refine the scope of the USV scheduling problem and model to be considered. The model is mid-level, in particular, we are not concerned with how to decide which targets are significant (high-level), nor are we concerned with the detailed engineering design of the vessels themselves (low-level). We explain with a functional architecture to identify the inputs, outputs, and assumptions relevant to the unmanned surface vessel observation-planning problem (USVOPP) model. Then we address the research classification of the USVOPP by comparing it to other problems in the literature such as the selective traveling salesman problem and team orienteering problem.

**Chapter 4 – Problem Formulation.** In this chapter, we develop the mathematical formulation of the USVOPP. We propose a combinatorial optimization model for the USVOPP.

In this chapter, we propose a mathematical formulation to the USVOPP based on orienteering problem models in the literature adapted to meet the constraints and requirements of the UVSOPP. We explain what data is required and how we converted the real-world scenarios into a mathematical representation useful for our tests. In addition, we present the mathematical notation and mixed-integer programming formulation of the USVOPP. After the mixed integer formulation, we review the various heuristics that could solve the USVOPP. The fourth section describes the various heuristics we developed to solve the theoretical USVOPP implemented.

**Chapter 5 – Results and Analysis.** This chapter covers the tests we have run on our model and reports the results and analysis of the tests. It establishes our objectives in the testing, provides the hypotheses that we test, describes the metrics used to discriminate among plans, and presents the test bed used for experimentation. We compare the results of our developed heuristic against a basic greedy construction heuristic and in small cases, with the optimal solution.

**Chapter 6 – Conclusion.** This chapter summarizes our work and findings. We discuss the effectiveness of our heuristic formulation, present suggestions for the modification of our algorithm and provide suggestions for future work in regards to USVs and team orienteering problems like the USVOPP in general.

## 1.2 Contributions

This research makes the following contributions:

1. A mixed-integer programming formulation for the USVOPP
2. The development and implementation of the optimal formulation implemented in ILOG OPL Studio 5.0
3. The development and implementation of a new greedy construction heuristic to solve the USVOPP and serve as a baseline to test against
4. The development and implementation of a new three-phase heuristic algorithm to solve the USVOPP implemented in Java
5. The development of a series of test datasets for hurricane and harmful algal bloom scenarios for the USVOPP

6. Experimentation and comparison of the implemented formulations
7. Recommendations for future modifications to solve the USVOPP and similar problems in general

The formulations in this thesis can serve as the role of an automated mid-level planner for a fleet of USVs. Specifically, this thesis proposes a new three-phase local search algorithm that exploits the unique structure of the problem to run quickly and efficiently for large problems. To the best of our knowledge, this is the first application of a local search heuristic to a variant of the team orienteering problem with time windows problem. The testing of the two scenarios provides insight into the quality of the algorithms we developed. The advancements made in this research could aid in automated fleet routing problems that could arise in future real-world observation applications.

# Chapter 2

# Observing Dynamic Oceanic Phenomena Operational Description

In this chapter, we present the problem of autonomous coordinated observation of dynamic oceanographic phenomena to predict and observe events such as hurricanes and algal blooms. The first section describes ocean phenomena in general, two significant example observation scenarios, and how each phenomenon is currently observed. The second section describes the physical components of the sensor web that enable enhanced observation methods. The third section describes the two scenarios for the sensor web. Finally, the fourth section describes the specific problem of unmanned surface vessel fleet path and observation planning that this thesis addresses.

## 2.1 Oceanic Phenomena Background

The ocean plays a major role in the world's climate and economy. For hundreds of years, scientists, mainly physical oceanographers, collected data to study oceanic phenomena. The oceanic phenomena include, "Interaction of the ocean with the atmosphere, how the ocean stores and releases heat, the physical properties of water throughout the ocean, and the formation and movement of currents and coastal dynamics" [53]. Currently, buoys, balloons, and satellites collect the majority of the world's oceanic data. "With satellite data, scientists can understand not only how the ocean behaves at a given point in time, but also how the ocean changes and fluctuates. For example, the patterns of heat distribution within the ocean and the geographic extent of current systems affect climate and weather" [53]. We can use this data to make models that more accurately forecast weather patterns. However, when dense clouds cover a region, satellite imagery instruments such as NASA's Moderate-resolution Imaging Spectroradiometer (MODIS) on NASA's Aqua and Terra satellites cannot produce pictures of the land and water. Therefore, in these regions MODIS cannot collect useful data. Recently, NASA developed the Advanced Microwave Scanning Radiometer (AMSR-E), a sensor on Aqua that can penetrate most types of clouds. However, the image resolution is lower than the thermal sensor operating in MODIS. Other means to collect MODIS-quality data over cloud-covered regions include utilizing unmanned aerial vehicles (UAVs), NASA's Ocean-Atmosphere Sensor Integration System (OASIS) unmanned surface vessel platform, or the Department of Defense Spartan Scout unmanned surface vessel.

The specific unmanned surface vessels (USVs) of interest are the OASIS USVs and the United States Navy Spartan Scout. The OASIS platform is an autonomous oceangoing surface vessel currently in development by National Oceanic and Atmospheric Administration (NOAA) in coordination with NASA. The Navy's Spartan Scout is a program aimed at developing and demonstrating a USV capable of performing various missions such as submarine reconnaissance and anti-submarine. USVs provide unique observation abilities. One important ability of the OASIS boats is the ability to act autonomously up to 60 continuous days in the water, while traveling at 2 knots. The most capable UAVs, such as NASA's Global Hawk, only have a flight time range of 42 continuous hours. The Spartan Scout acts more like a UAV and only has 5 hours of endurance when traveling at top speed. At lower speed, the Spartan Scout can travel up

to 100 hours continuously. The Spartan Scout does have the ability to travel at speeds up to 40 knots through the water. Either of these USVs is less expensive than an UAV; OASIS USVs are estimated to cost less than $20,000. A major advantage of USVs is that they can collect in-situ measurements from the water. The vessels can take samples directly from the water, collect accurate temperatures up to several feet below the surface, and have the ability to collect those samples in adverse weather conditions, unlike UAVs.

NASA is developing the Adaptive Sensor Fleet (ASF) system. The ASF system is a proposed control software package in development for application to a wide array of sensor-equipped autonomous vehicles. The software is designed so that a user can adapt the system to the dynamics of various types of vehicles and their operating environment. We will take this idea of an ASF to develop a coordinated USV Planner. We focus on using the USV Planner to coordinate USVs to observe two oceanic phenomena that cause major problems for the United States and the world: harmful algal blooms (HABs) and hurricanes. The USVs can collect in-situ data used to identify the occurrence of harmful algal blooms and predict a hurricane's path and intensity. Both of these phenomena could be better understood if more accurate in-situ measurement capabilities existed.

### 2.1.1 Harmful Algal Bloom Identification and Monitoring

Marine life is important as a source of food for the United States and the world. Algae form the base of the marine life food chain. Under certain conditions, millions of algae concentrate together, produce and release toxins, and cause a Harmful Algal Bloom (HAB). HABs can have detrimental impacts of major concern. Certain naturally occurring plant organism toxins can cause severe consequences for marine life, human health, the environment, and the economy. The National Science and Technology Council Committee reported that, "Harmful algal blooms threaten human health and natural resources throughout US coastal waters, from Alaska to the Gulf of Maine" [33]. The Woods Hole Oceanographic Institute estimated that, "...The annual aggregate economic impact (in millions of 2000 dollars) of HABs in the United States during the 1987-1992 period... average $49 million per year, ranging from $34 million to $82 million. Over the last several decades, the cumulative impacts thus approach $1 billion" [24]. An important objective is to reduce the negative impact with better

identification techniques and more accurate data collection on the movement of HABs. This will enable scientists to produce forecasts that are more precise and mitigate problems in the future.

Most types of algae in the Earth's coastal waters and oceans are harmless. Out of several thousand species of phytoplankton (cyanobacteria) only a few species are known to be harmful. "Certain species of phytoplankton (e.g. Alexandrium, Gambierdiscus, Prorocentrum) are known to produce toxins that are harmful, and potentially lethal, to marine life and to humans" [59]. Pfiesteria is another; it thrives in polluted waters and has caused numerous fish kills in the Mid-Atlantic States. Some of the algae have been found to be responsible for the human bacterial disease cholera [49]. It is difficult to distinguish between the harmful and harmless types of algae, but with in-situ measurement, it is possible. HABs occur in high water temperature areas with low salinity, high sunlight, little wind, and minimal mixing of the water. Areas experiencing these conditions are ripe for intense HABs and are high-value areas to monitor actively.

Today, the method for HAB detection is periodic monitoring. Scientists use Global Information Systems (GIS) to map and observe HABs, but initial notification and response involves a chain of events that are based largely upon human involvement and interaction. Scientists routinely monitor marine environments thought to contain conditions conducive to HAB outbreaks. Additionally, regions that are commercially fished are actively monitored by scientists working for the Department of Natural Resources or hired by the commercial fishermen. This process is inefficient in the utilization of time and people. "There are also inefficiencies associated with the time (and therefore the cost) for the researchers that are required to travel to the location(s); take measurements and retrieve samples; record information (e.g. time, location); return samples to the lab; and then test, analyze, and document findings" [59]. When a scientist identifies an HAB, a subsequent chain-reaction of events occurs to lessen its negative impact. The scientist first notifies the environmental protection organizations such as the National Center for Coastal Ocean Science and the Environmental Protection Agency; if resources are available, the organizations travel to the region, collect water samples, perform the in-situ analysis, confirm the toxicity concern, map the characteristics of the HAB, and predict the future growth and dispersion. Because the process is very time consuming, scientists have placed buoys to collect continuous data in some of the areas of interest. However, buoys cannot move, making it difficult to track HABs, and need routine maintenance when operated over a

long period of time. Many buoys could offer good data from an area, but additional buoys provide minimal benefit for the additional cost.

### 2.1.2 Hurricane Path and Intensity Prediction

Hurricanes have had significant impact on life and prosperity. Hurricane Andrew (1992) and Hurricane Katrina (2005) devastated parts of the United States. Scientists have suggested that global warming has contributed to an upward trend in stronger hurricanes over the past 30 years. Massachusetts Institute of Technology Professor Kerry Emanuel of the Program in Atmospheres, Oceans, and Climate stated that, "The energy released by the average hurricane (again considering all hurricanes worldwide) seems to have increased by around 70% in the past 30 years or so, corresponding to about a 15% increase in the maximum wind speed and a 60% increase in storm lifetime," [23]. Stronger hurricanes have greater economic impact. The Annual Meetings of the American Economic Association estimated, "...That the average annual U.S. hurricane damages will increase by $8 billion at 2005 incomes (0.06 percent of GDP) due to global warming. However, this number may be underestimated by current storm models," [46]. An objective is to mitigate the effect of hurricanes by collecting more data used to predict more accurately the future paths and the impact on coastal locations affects from hurricanes.

Weather prediction, including the prediction of hurricane path and intensity, is a difficult and often inaccurate science. Billions of dollars have been spent to make the prediction process better, and it has improved to some extent. However forecasting hurricanes is still problematical. Currently, scientists use satellite data, specially equipped aircraft data, and hurricane history to predict the path, landfall location, and intensity of a hurricane. NOAA maintains a network of buoy and Coastal-Marine Automated Network (C-MAN) stations. This network provides forecasters with marine measurements related to wind conditions, barometric pressure, temperature (air and sea surface), and wave data. This data is transmitted to the National Hurricane Center in Miami and the National Centers for Environmental Prediction in Camp Springs, Md. During the past several years, the addition of the Gulfstream-IV aircraft has increased data collection capabilities by flying in and around each hurricane. "G-IV flight data are expected to help numerical guidance computer models improve hurricane landfall and observe forecasts by up to 20 percent, and to further refine storm intensity forecasts," [45].

Professor Kerry Emanuel stated that, "If the average wind speeds near the surface of the tropical oceans does not change; theory predicts that the wind speeds in hurricane should increase about 5% for every 1°C increase in tropical ocean temperature," [23]. Observations taken of the surface of the ocean will provide estimates of average wind speeds. To forecast hurricanes to the same level of accuracy as the forecasting of storms across the United States, more observations are needed. University of South Carolina marine biologist Madelyn Fletcher said, "The density of U.S. coastal weather observations is relatively low compared to those over land. Only 140 sites collect data along the coasts compared to 14,000 on land. A lot of our weather comes from the ocean, that emphasizes the great need to have more measurements, more observations, and the serious need for a higher density of observations sites in coastal oceans," [43]. Data from in-situ measurements from the water can increase the accuracy of forecasts, provide calibration and validation of aircraft and satellite data, and increase the overall resolution of observations. In order to collect the data required, additional and more sophisticated data collection methods are needed.

## 2.2 Sensor Web Description

In order to increase the amount of observation data gathered effectively, we will use the innovative idea of a sensor web. NASA's Stephen Talabac characterizes a sensor web as follows:

> A sensor web is a distributed, organized system of nodes, interconnected by a communications fabric that behaves as a single, coherent instrument. The exchange of measurement data and other information, produced and consumed by its sensing and non-sensing nodes, causes the sensor web to dynamically react by appropriately modifying subsequent sensor measurements and reconfiguring node information processing states in ways that tend to optimize useful science return [59].

A sensor web enables more refined and effective methods to monitor the dynamic behavior of events and phenomena. These methods incorporate interconnected resources that can coordinate autonomously and reconfigure themselves in a manner that passive observation systems in use today cannot generally do. The sensor web's resources exist on or below the surface of the Earth, within the atmosphere, and in space. Each resource either contributes measurements and

information from its sensors, or supplies computing and storage capabilities to complement the sensors.

A sensor web encompasses more than just a distributed data collection system. Distributed data collection processes have existed for many years. A distributed data collection system has multiple distributed nodes, such as ground, air, and space resources that make measurements and report raw sensor data to a central site where the measurement data is collected. The central site then formats, calibrates, and combines the data to enable higher-level decisions based on the processed data. But in a sensor web, the data flows in multiple directions. The distributed data collection system sends data from the sensor nodes to the processing nodes. A sensor web interchanges information among sensor nodes, and data processing nodes. A sensor web system enables dynamic reaction to sensor measurements, events, and state changes of the other nodes in the web. An illustration of a sensor web is shown in Figure 2.1. The interconnected dotted lines between the different vantage points represent the flow of data from one data collection node to another. Distributed data collection systems only send data to collection points in the system.

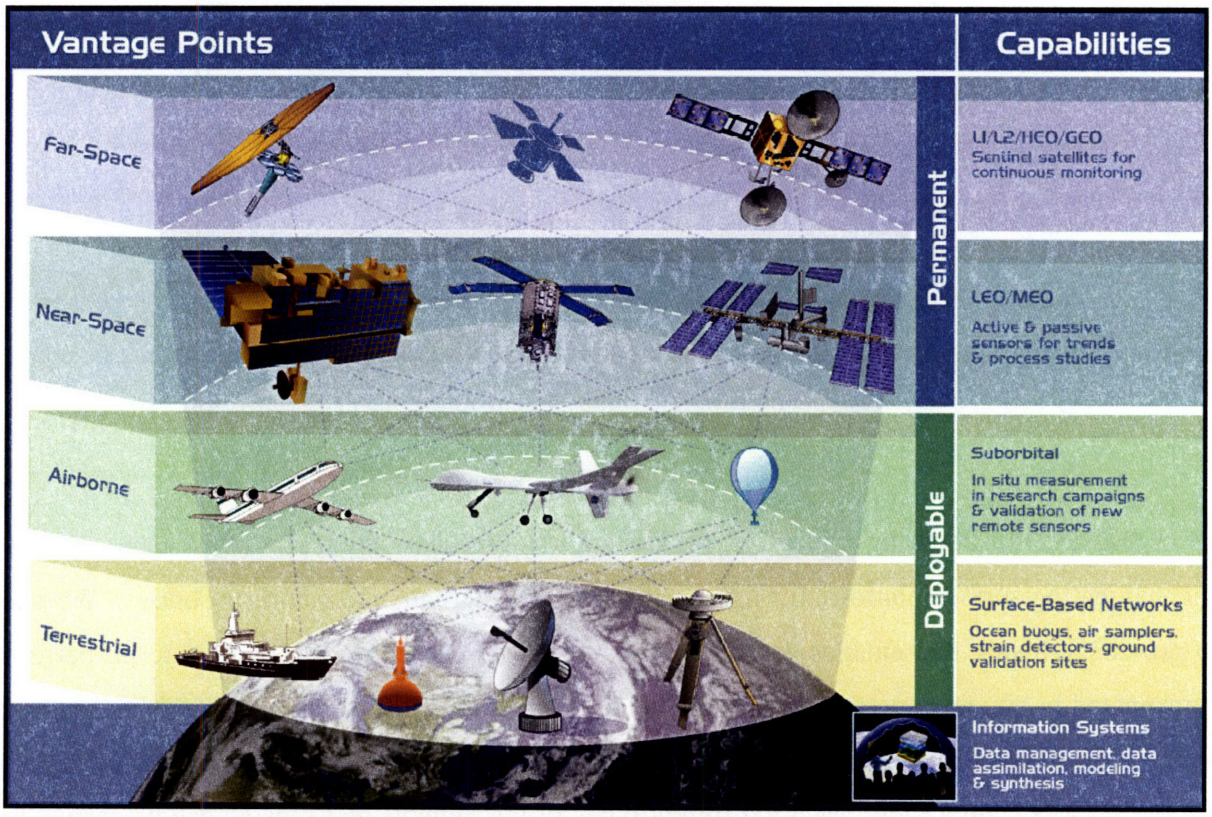



**Figure 2.1: Sensor Web Diagram** [59]

### 2.2.1 Unmanned Surface Vessels Operational Description

We focus on the role that USVs will play in an overall sensor web. The idea stems from the envisioned mission of the Adaptive Sensor Fleet (ASF), which is, "To provide the capability for autonomous cooperative survey and sampling of dynamic oceanographic phenomena such as ocean current systems and algae blooms," [55]. Our goal for our research is to enable USVs to perform coordinated observations and data collection of a dynamic environment optimally. The design of the OASIS USV and the Spartan Scout enables them to be used as part of a dynamic sensor web. Many features of the OASIS USV and the Spartan Scout make them ideal for data collection, calibration, and validation of remote sensing satellite measurements.

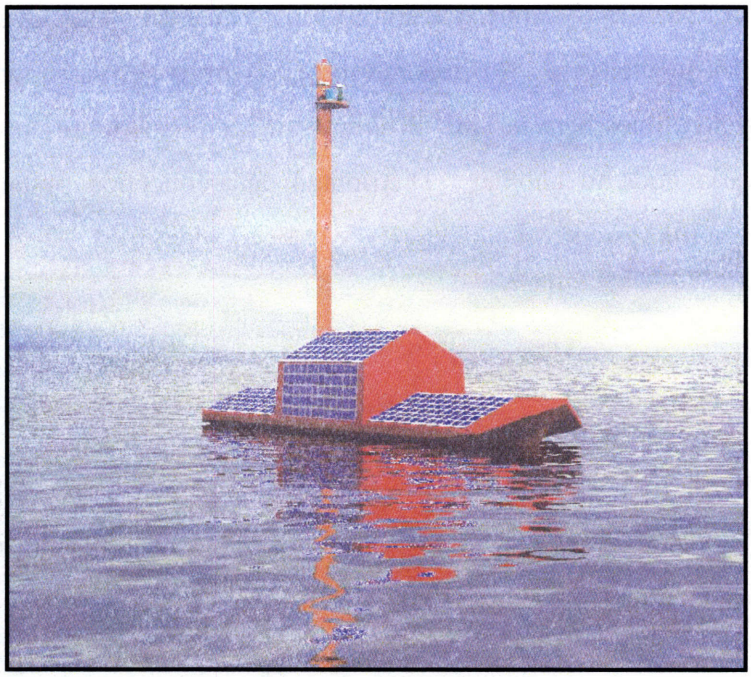


**Figure 2.2: OASIS Platform** [16]

OASIS, illustrated in Figure 2.2, provides a low-cost (less than $20,000 dollars), mobile, self-navigating surface platform for ocean sensors as an alternative to stationary buoys. We plan to provide software that enables the USVs to navigate to different locations in order to perform coordinated in-situ measurements, and then return home to recalibrate and replace instruments if necessary. This will provide scientists the ability to map and observe phenomena in regions remote sensors cannot view. Many different types of instruments and sensors can be carried by the OASIS platform allowing diverse scientific experiments. In this thesis, we will focus on the

surface temperature sensor, below surface temperature sensor, and the Prototype Phytoplankton Fluorescence Sensing System (PPFSS) Fluorometer. These sensors provide data for HAB detection and hurricane path and intensity prediction.

Another USV that could potentially perform similar functions as the OASIS USV is the United States Navy's Spartan Scout. The Scout can perform military missions such as anti-submarine, mine countermeasures, surveillance, reconnaissance, and precision strike depending on the attached equipment and weapons. We are assuming for this research that we could utilize the OASIS platform equipment on the Spartan Scout. Figure 2.3 illustrates the Spartan Scout USV.


**Figure 2.3: Spartan Scout Picture** [34]

The USV observation planner that we will develop will formulate a quick and efficient configuration of the USVs for conducting the observations based on current environmental conditions and USV characteristics. Our algorithm will efficiently coordinate the USVs using a local search heuristic. Users of the USVs will be able to update observation goals in real-time based on the data the USVs collect, or on changes in the environment.

## 2.3 Scenario Descriptions

We now describe how a future sensor web that includes a fleet of USVs could be utilized in the harmful algal bloom and hurricane scenarios and discuss the advantages of a sensor web for these scenarios.

### 2.3.1 HAB Monitoring Scenario

The following scenario is an illustration of how a reconfigurable sensor web could be used to identify, locate, analyze, and observe a harmful algal bloom. This process could improve the current ability to collect data, observe, and forecast the evolution of a harmful algal bloom outbreak. The use of existing Earth observing satellites and other sensor systems is notional.

1. The MODIS sensor onboard NASA's Aqua EOS (Earth Observing System) satellite produces data images of the ocean color and temperature from space. On a routine pass over the United States, MODIS produces photographic images of various areas from the Gulf of Mexico and the Atlantic Ocean off the eastern shoreline of the United States.
2. During ground processing of one of the images from MODIS, a scanning algorithm identifies a potentially high concentration of chlorophyll within the image in the Chesapeake Bay region by identifying a discolored region in the water and irregular temperature readings in the area. HABs, among other phenomena, can cause this high chlorophyll concentration. The coordinates of the region are sent to another satellite carrying MODIS, such as Terra, to specifically target and image the area for HAB discrimination. If available in the region, the ground station sends a warning to the USV observation planner and to an Altus II unmanned aerial vehicle from the Goddard Space flight center via an onboard UHF/VHF omnidirectional antenna.
3. The ground station receives the image and also sends a warning that a possible HAB has developed to the automated science goal analyzer for the sensor web. The science goal analyzer calculates the probability that an HAB has developed from the given information in the image. If this probability is 95% or higher, the science goal analyzer sends a warning to the coastal observation service in Virginia of the HAB outbreak and

alerts the USV observation planner of the current size and location of the HAB. The USV observation planner is the automated controller of a fleet of USVs. If the USV planner is given an order by the science goal analyzer, it efficiently routes the USVs to collect data and/or observe the HAB outbreak. The science goal analyzer also passes on current weather patterns, so the USVs can collect specific data about potential harmful effects from the HAB and observe the expansion of the possible HAB. If the probability is between 75% and 95%, the science goal analyzer orders the USV observation planner to observe the area and collect in-situ measurements to determine if the outbreak is truly harmful. If the probability is between 50% and 75 %, the science goal analyzer either orders a NASA UAV already in the air to observe the specific coordinates or schedules a new flight to observe the specific coordinates under the cloud cover. If the probability is less than 50%, it schedules instruments on trailing satellites to observe the coordinates again.

4. If the probability is in range that calls for USV observation, but the outbreak is found to be too far away from any of the USVs, then the station automatically sends notification to commercial fishing boats and the public that a potential HAB is in the specified area. Otherwise, it waits for resources to reach the potential HAB area and uses UAV and satellite data until the ASF can collect the specific in-situ measurements necessary to accurately predict the ill effects and diagnose the cause of the outbreak.
5. Steps 1-4 create a cycle that continues until the HAB warning clears, or the HAB is identified and proper actions taken. The warning could be cleared by a UAV uploading images to the science goal analyzer, the USV uploading collected data readings, or satellite images uploaded to the analyzer. If an HAB is identified and USVs are observing an HAB outbreak, the USVs provide continuous feedback on the size, location, and movements of the HAB. The feedback is sent to the science goal analyzer and to the Coastal Observation network. Once the possible ill effects of the HAB have been determined, the information is sent to the emergency response personnel. The coastal observation network and emergency personnel notify the fishing industry and the public of the areas to avoid and of any potential health risks.
6. The USVs, UAVs, and satellites continue to monitor the outbreak area until the HAB is nontoxic. Data gathered by the three sensor systems in the area is collected to validate

the remote sensing data against the in-situ measurements from the USVs. The data is also used to improve methods for future forecasting of HAB outbreaks.

7. The warnings and improvements from the system ensure that commercial fishermen do not fish the harmful areas and swimmers avoid toxic areas. End result: a minimization of a potentially costly and unhealthy threat.

The scenario can start with any sensor; OASIS may find the first sign on a routine pass, or any of the other sensors such as a UAV, a satellite, or a human detection node could start the cycle. In any case, the information is shared with the nodes of the sensor web that need to know and the scientific analyzer determines how to break up the next assignments among the gathering sources. We plan to take this overall sensor web scenario and analyze how the fleet manager of the ASF can optimize the OASIS vessels.

For the purpose of this thesis, we assume that the HAB has already been identified. We then take on the role of the fleet manager and assign the boats to work together to monitor the expansion and growth of the HAB. A series of pictures of a scenario where an HAB was identified in the Chesapeake Bay off Virginia follows:

**Figure 2.4: Time Snap Shots of HAB Identification Scenario**

The first snapshot shows the initial location of six boats when a potential HAB is identified by a satellite image. The blue dots represent the six boats; the red oval represents the outline of the HAB. The boats in the region are all called to the scene to observe the growth of the bloom. In the second snapshot, the boats move toward the scene. The nearest boat has already started mapping the southern edge of the bloom. In the third scene, four more boats reach the edge of the bloom and start observing the bloom on each side. The final scene depicts the boats following the algal bloom until it dissipates.

### 2.3.2 Hurricane Intensity and Path Prediction Scenario

The second scenario describes how a notional reconfigurable sensor web might be used to improve prediction of the future path and intensity of a hurricane approaching the United States.

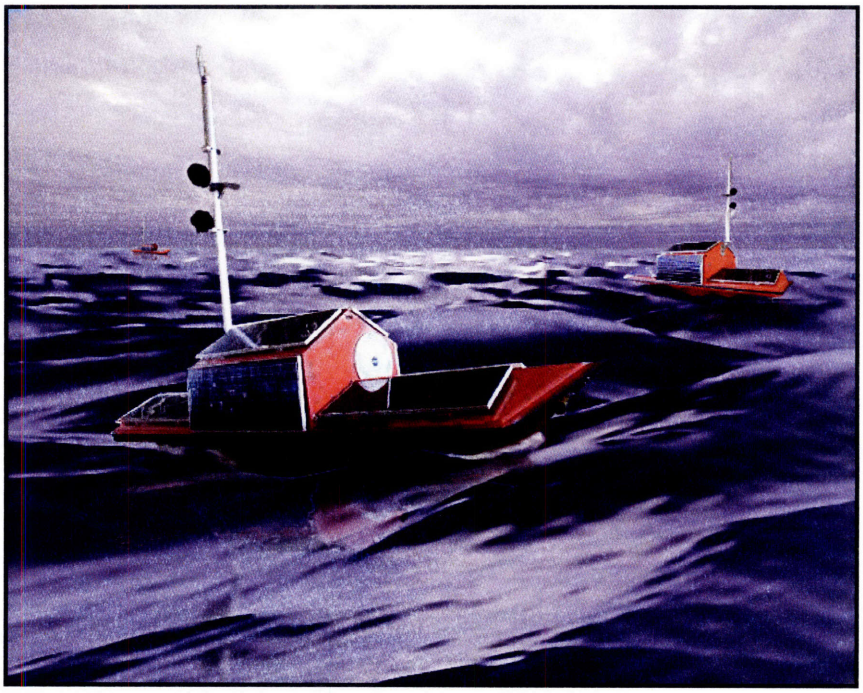


**Figure 2.5 Coordinated Hurricane Observation Picture** [60]

1. One of NASA's satellites in low Earth orbit, Aqua, using the MODIS sensor, produces data images of the ocean temperature from space. On a routine pass over the Caribbean Sea, the satellite produces photographic images from the predicted path of a newly formed tropical storm.
2. Within one of the images, a science survey algorithm identifies high water temperatures in the Atlantic Ocean off the coast of Florida in the current projected path of the tropical storm. The science survey algorithm also finds wind speeds topping the hurricane level and immediately identifies the storm as a hurricane. The science survey algorithm sends a warning to NOAA's "Hurricane Hunter" aircraft with any new information it gathers and the hurricane coordinates. It also sends a warning to the USV observation planner if the USVs are in the region. Finally, the satellite downloads the image to the ground

station to alert the National Weather Service (NWS) and National Hurricane Center (NHC) for further analysis.

3. The "Hurricane Hunter" aircraft flies to the scene of the new hurricane and scientists aboard the aircraft deploy instruments called GPS (Global Positioning System) dropwindsondes. These devices continuously radio back to the USVs, NWS, and NHC, measurements of pressure, humidity, temperature, wind direction, and speed as a function of position and altitude, as they fall toward the sea, which provide a detailed look at the structure of the storm and its intensity.
4. The USV observation planner deploys the USVs in the region to the area in the path of the hurricane. The planner efficiently routes the USVs to collect data ahead of the storm that aircraft, satellites, and buoys cannot collect. Because of the harsh weather conditions, the USVs provide a unique and novel data collection capability. As they collect the data, the USVs upload the water temperature, surface wind speeds, and water currents to the other resources in the sensor web.
5. The data is analyzed, and the NHC issues more detailed and accurate path predictions, evacuation orders, and intensity readings taken from the sensor web. Meanwhile, the sensor web keeps working together to provide up-to-the-minute readings and forecasts. The sensor web will also call in more resources if deemed necessary.
6. The USVs, UAVs, and satellites continue to monitor the critical region until the hurricane has left the region or no longer threatens land. Data gathered by the three sensor systems of the area is collected to validate remote sensing data against the actual in-situ measurements from the USVs.
7. End users of the USVs and the sensor web use the information to enhance forecast models and better predict future hurricane paths and intensities. The better predictions enable the citizens in the landing zone to be evacuated and noticed in record times, and the avoidance of evacuating citizens not in harms way.

In our research, we assumed that a hurricane has been identified, and the OASIS USVs in the region have been directed to the region. We then take on the role of the USV planner and assign the boats to work together to collect data ahead of the hurricane in order to optimize the

prediction of the path and intensity of the storm. Figure 2.6 illustrates the search region in a scenario where a hurricane is approaching Florida:

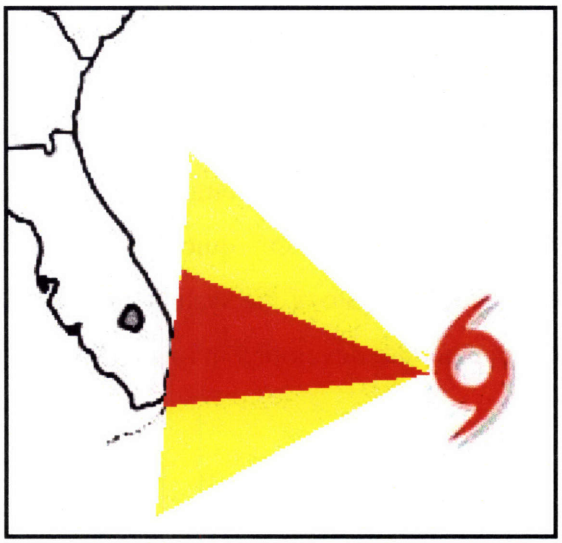


**Figure 2.6: Hurricane Scenario Diagram**

The red area in the figure shows the most important region in determining the intensity and path of the hurricane. The yellow region represents the area with a less significant data value for the USVs to collect than the red region, but still valuable enough to explore.

## 2.4 Problem Statement

For each of the two scenarios, our problem for the USV planner is very similar. Consider the HAB scenario in section 2.3. Somewhere in the sea is an algal bloom. A USV planner customer suspects the feature is out there, but does not know the precise location or extent. Therefore, the customer identifies a polygonal region of interest, and requests an USV fleet to observe it. The USVs, beginning from arbitrary initial positions, are tasked to travel quickly to the region, and then observe it in an efficient manner. In transit, there are land features, and other navigational hazards, that must be avoided. Water currents exist, that need to be accounted for, as they affect travel time of the USVs. In the region, the observations must be assigned to individual USVs based on initial locations, travel times, and the value of different locations in the region.

In a similar manner, the USVs react to a customer concerned with the hurricane scenario. Instead of observing the spread of the bloom as in the HAB scenario, the USVs coordinate to gather data observations from the water that improve hurricane path prediction. The challenge for both of these scenarios is how to coordinate multiple USVs to observe many thousands of target locations with time sensitive observation values. Optimized coordination of many vessels observing many targets has the potential to require massive amounts of computing power and time. This problem requires dynamic solutions computed in under a few minutes. We need algorithms that solve the coordination problem quickly so that the USVs observation and movement plan evolve over time as the state of the system changes. Algorithms that accomplish these tasks in a time efficient manner are developed in this thesis.

# Chapter 3

# Model Development

As was depicted in the section on the notional Sensor Web (Section 2.2), many different elements of a sensor web work together to monitor the dynamic behavior of events and phenomena. The rest of this thesis focuses on observation planning for unmanned surface vessels. As the previous chapter presented the operational background, this chapter explains the development of our model by comparing our problem to other problems in the literature. The model is mid-level, in particular, we are not concerned with how to decide which targets are significant, nor are we concerned with the detailed engineering design of the vessels themselves. This chapter begins with a functional architecture to identify the inputs, outputs, and assumptions relevant to the model. Then we address the research classification through a literature review of similar problems.

## 3.1 Functional Architecture

The focus of our research is guided by our estimation of USV utilization. We envision that the USVs would be dispersed along the coastal areas observing a collection of various targets to aid in normal operational data collection for an overarching sensor web. The overarching sensor web consists of space-based sensors, airborne sensors, and surface-based sensors. Figure 3.1 illustrates the coordination of the different resources working together under a sensor web observation coordination planner.

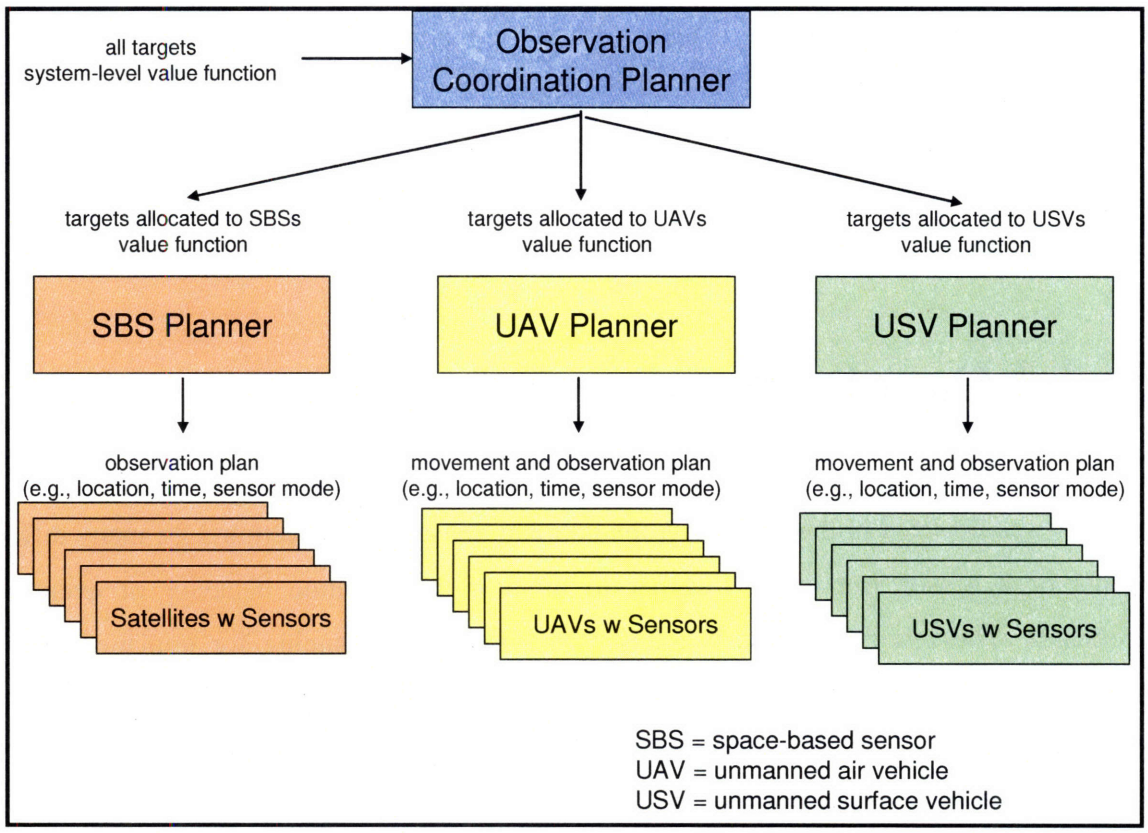



**Figure 3.1: Functional Architecture for Sensor Web Planning**

The sensor web observation coordination planner allocates all the targets deemed significant to observe optimally among the space, airborne, and surface planners. Depending on the resolution demanded and the type of target, one or more mid-level planners could receive the same target. After the targets have been allocated among the different planners, each planner adds the new target list from the Observation Coordination Planner to its ongoing observation target list. After

combining the target lists, each planner dynamically reformulates how to utilize its resources effectively for the new problem. For the USV Planner, target lists include points where measurements of water temperatures, chlorophyll levels, and surface wind speeds are required. In addition, the target lists include points that could verify and validate other devices by collecting data in the same location as a satellite, UAV or a buoy.

### 3.1.1 USV Observation Planning Overview

To optimally plan and coordinate USV observations, there are many different decisions of the overall process to consider. For the purposes of this research, we generalize the process of USV Observation Planning into three levels of decisions: high-level, mid-level, and low-level. The overall process encompasses the time from target identification to data transmission of the target collected. The high-level decisions of the process include the establishment of objectives and development of target lists to accomplish these objectives. The mid-level decisions include assigning orders for the USVs by producing a schedule for observation planning for each USV. The low-level orders include how the boat physically travels, transmits information, and receives information. Examples of the types of questions for each level:

High-Level Decision Examples:

- What are our objectives?
- How many vessels do we utilize?
- Which events are important to observe?
- What targets are significant?
- What targets have priority?
- What is the planning horizon?
- What instruments do we use?

Mid-Level Decision Examples:

- How do we assign boats efficiently, maximize value, and meet objectives?
- How do we satisfy the constraints for the problem?
- How well does the plan satisfy our goals?

Low-Level Decision Examples:

- How does the boat observe a target ?
- How does the boat download instructions?
- How does the boat upload data?
- How does the boat move from one target to another?
- What trajectory is optimal?

We focus our research to answering the mid-level questions. We assume that the high-level decisions are decided by the overarching Observation Coordination Planner. In addition, we assume that the boats have the required equipment and procedures to collect data, travel in the water, transmit and download information, and find a trajectory to travel point to point, all low-level decisions. With these assumptions, we can successfully confine the scope of our research to explore the mid-level decisions of the USV Planner. Because USVs have a very long endurance, USVs can perform several tasks and visit many targets in a specified time period. This fact makes observation planning for these USVs a good candidate for optimization, as improved observation assignments among USVs could mean more tasks accomplished and scientific value attained over each planning period. However, when we apply an optimization method to a problem that has uncertainty, such as this problem, we must model carefully to return good results. We need to decide how to translate our physical problem into a model that represents the real-world problem well and that we can easily analyze.

### 3.1.2 USV Observation-Planning Problem

This section describes the USV Observation-Planning Problem (USVOPP) and defines the inputs and outputs of the problem. It presents the assumptions that we make on the capabilities of the USVs, and explains any other assumptions that we make in the model. For simplicity, we include only two types of USVs, the Spartan Scout, and the OASIS. The Spartan, which is faster, has a higher range in a given time, but has less duration. OASIS, which is slower, has minimal range in a given time, but has greater duration.

### 3.1.2.1 Characteristics of the USVOPP

At the initialization of the USVOPP, we are given an initial configuration of USVs together with a list of targets. The location, value, and time window to be observed of each target is specified. In addition, the travel times are computed with a time-variant method to account for tidal currents. We wish to create an observation plan for the USVs in a way that maximizes the total value of the targets observed in a specified length of time. The objective of the USVOPP is to maximize the total value collected by the USVs from the targets.

### 3.1.2.2 Inputs

The inputs to the USVOPP model include:

**Model Inputs:**

1. Number of USVs available
2. Number of targets
3. Coordinate locations of USV starting locations and targets
4. Velocity and direction of the tidal current
5. Velocity of each USV
6. Required target observation times
7. Target observation values
8. Time windows of opportunity for each target to be observed
9. Planning horizon

The data inputs are used to define a particular scenario and are required to initialize our model. The number of USVs available and the number of targets define the size of the problem. The coordinate locations of USV starting locations and targets, the velocity and direction of the current, and the velocity of each USV define the travel times for the problem. The required observation times define the time a USV spends stabilized to accomplish a particular task at a target. The times are specified by target and task required. The target observation values define the target's significance to the overall science goals when observed. In addition, the scientific

value is a positive constant if the target is observed during an inputted particular interval of time, called the time window of opportunity, and zero otherwise. The last input to the USVOPP is the specific operating planning horizon that defines the ending time for an observation plan. All of the data inputs we mention are mean values, which are subject to uncertainty. The data inputs for the model are representative of what we believe to be logical values for estimation.

#### 3.1.2.3 Outputs

The outputs to the USVOPP model include:

**Model Outputs:**

1. Inter-target observation path used in plan for each USV
2. Finished observation times for each location observed
3. Idle times at each leg for each USV
4. Total value for each USV plan

The output from the USVOPP is an observation plan, which schedules the USVs and assigns the tasks required of each USV. The plans specify the paths that the USVs follow through their assigned tasks, the finished observation time of each boat at each location on the route, and the idle time for each USV at each location. The routes are defined by legs that correspond to the inter-target paths selected by the automated planner on which the USVs travel between targets. The plan also includes the valuation of each route and total value for all USVs.

#### 3.1.2.4 Assumptions

The assumptions to the USVOPP model include:

**Model Assumptions:**

1. Two types of USVs, Spartan Scout and OASIS
2. The speed and endurance are the only differences between the two types of USVs

3. OASIS USVs travel at a constant 2 knots, Spartan Scout USVs travel at any constant speed less than a maximum speed of 40 knots
4. OASIS has an endurance of 60 days; Spartan Scout has an endurance that changes polynomially with its speed as illustrated in Figure 3.2
5. No USVs can be lost or damaged
6. Temperature readings can be completed in route, i.e. observation times at each target are zero
7. Time to collect a fluorometer reading requires a complete stop, and 1 minute of time to observe the target
8. Routes are open ended, i.e. no designated finish location
9. Targets may or may not be observed, but value is received for successful observations
10. Inter-target travel times are approximated by dividing straight-line distance between the targets by current-adjusted speed of the USV
11. Currents in the Chesapeake Bay are tidal, and direction and magnitude at any location varies with time, in an approximated periodic way
12. Specific planning horizons range from a two hours up to 14 days

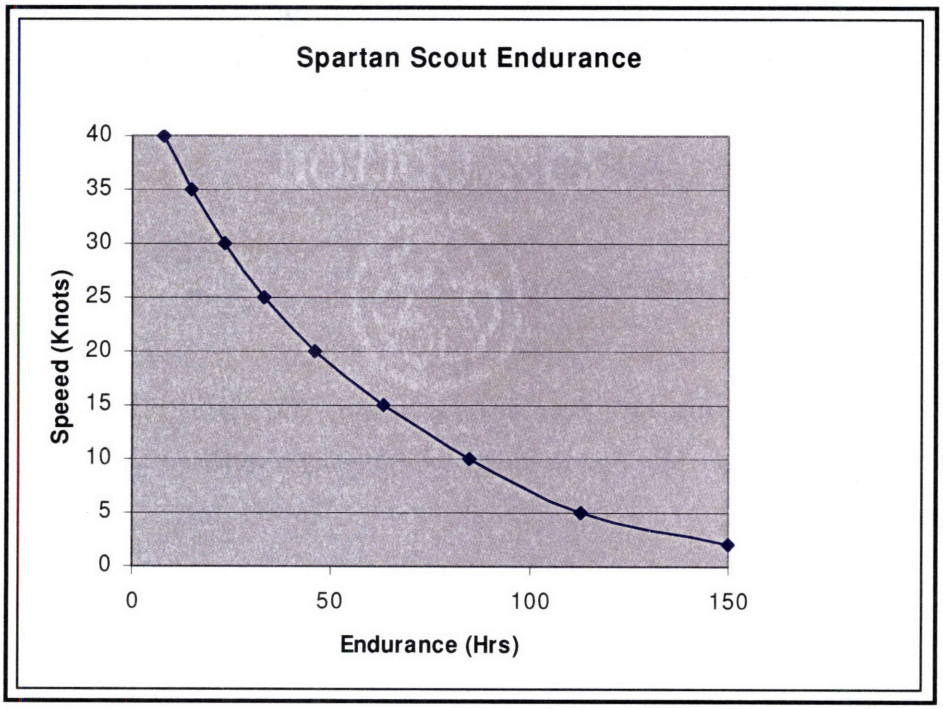



**Figure 3.2: Assumed Spartan Scout Endurance Graph**

For our model, we are focusing on the use of only the OASIS and Spartan Scout USVs. In addition, we assume that the only difference between the two USVs is the speed and endurance. This assumption infers that all tasks are completed homogenously for each type of USV. We assume the maximum speed of the OASIS USV is 2 knots, and the Spartan Scout is 40 knots. The range for the OASIS USV is up to 60 days and the Spartan Scout range is dependent on the speed, illustrated in Figure 3.2. In addition, we assume the time to record a temperature reading at a target does not require a USV to stop. This assumption is reasonable because a target represents a specific coordinate representing a square area dependent on the resolution. We assume that during the time the USV is moving to the specific coordinate in the center of the square of area, it can collect the water temperature-reading representative of that target. However, we assume collecting a fluorometer reading requires a full stop and 1 minute to collect the reading. This assumption gives us a different scenario where the vehicles have to stop at targets instead of collecting in route. In addition, we assume that USVs are never lost or damaged. This assumption simplifies our decision variables for the model because we do not have to account for a ship losing speed or becoming disabled.

We assume that the observation plans can end at any target, and do not have to include observing all the targets. Distances are approximated by dividing straight-line distance between locations. The travel time is approximated by dividing the distance by current-adjusted speed of the USV. We assume typical planning horizons range from a couple of hours up to a couple days. USVs begin a plan at any initial location.

## 3.2 Graphical Representation

This section describes how we transform the physical real-world problem into a graphical structure we can visualize. We first describe a simple static graph of the problem and then build into a time-space graph.

### 3.2.1 Static Graph Representation

Applying analytic techniques to evaluate the USVOPP requires that the physical system be mapped to the proper theoretical framework. Large-scale transportation problems, such as vehicle routing problems, whose structure is similar to the USVOPP, are often represented using a directed graph, $G(N, A)$, consisting of a set of nodes, $N$, and a set of arcs, $A$. There are many ways to translate the physical characteristics of the USVOPP into such a graph.

We present one possible directed graphical representation of the USVOPP. Nodes correspond to initial USV locations and target locations. Directed arcs represent permitted motions of the USVS between the nodes. The directed arcs between the nodes indicate the direction of the flow between the two nodes; the tail of the arc corresponds to the origin and the head of the arc corresponds to the destination. An example diagram of a static graph is in Figure 3.3.

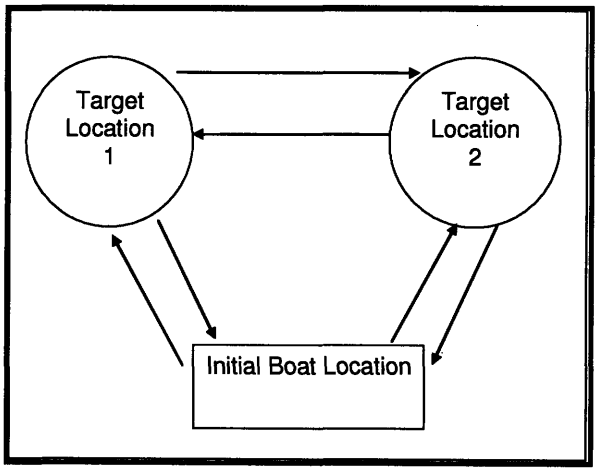



**Figure 3.3: Static Graph Representation**

### 3.2.2 Time-Space Graph Representation

The graph in Figure 3.3 is static, which means it does not allow USV movement to be modeled in the time dimension. The static graph is modified to create a time-space graph, which captures the USV movements through both the spatial and temporal dimensions. The time-space graph is constructed by creating a node for each location at each time in the planning horizon.

The planning interval is the entire time period of user interest. The planning horizon for the USVOPP can be any length of time, in hours or days, in which an oceanic phenomenon may last. Each node in the time-space graph is indexed by target and time period. For instance, the index $(1,t)$ corresponds to location 1 at time $t$. Directed arcs connect the nodes and represent USV flows in time and space. A time-space graph representation of the static graph in Figure 3.3 is illustrated in Figure 3.4.

An arc in the time-space graph that connects two nodes with the same location index means that the USV did not travel during that time period. This could be due to the USV observing a node, or the USV simply idling during that period. The rest of the arcs represent the USV traveling from one location to another location during that time period. This simple graphical representation does not distinguish between idling and observing at a location. It also does not account for time window constraints. However, it illustrates a good framework to visualize the problem.

The formulations in the remainder of the thesis use a modified time-space graph to visualize a coordinated observation plan. A plan contains the routing and execution details for the USV paths through the network of target nodes. The execution details make up the USV action at a particular target. Specifically, how long the USV waits, how long it collects data, and by what means it collects data.

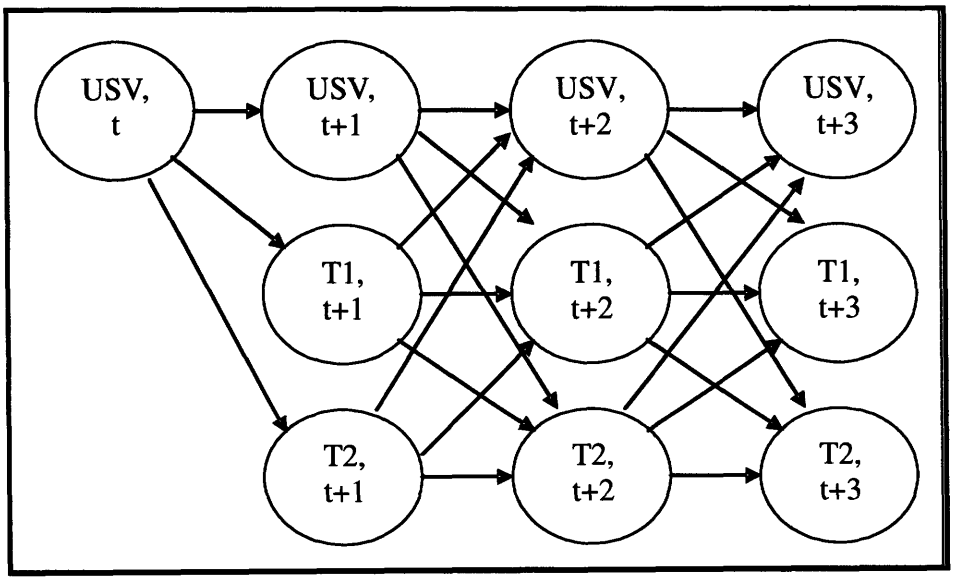



**Figure 3.4: Time-Space Graph Representation**

The time dimension in the Time-Space graph represents only a time period. The real-time is not modeled in the time-space graph. The real-time is tracked with a label on each arc, or arrow in the graph, of how long the vehicle spent on that arc in reaching the next location.

We show the advantage of this approach through a small example. If we assume that it takes the complete planning horizon for a USV to travel to Target Location 2, we can eliminate any arcs that cannot be reached within the specified planning horizon. Specifically, if the planning horizon is 8 hours, the travel times from the USV Location to Target Locations 1 and 2 are 4 and 8 hours respectively, and traveling from Target Location 1 to Target Location 2 is 6 hours, we can formulate a new graph with fewer nodes and arcs. Figure 3.5 illustrates the new graph for this example. The numbers next to the travel arcs represent the time in hours for an USV to traverse that arc. Idle arcs do not have a definite label and are determined by the USV Planner.

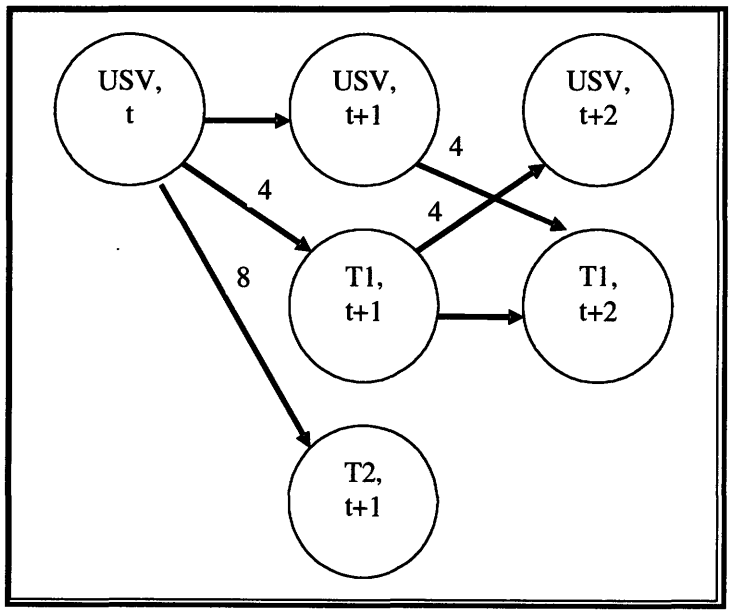



**Figure 3.5: Modified Graph for Small Example**

For this example, any location that would cause the USV to exceed 8 hours is eliminated. Smaller planning horizons reduce the size of the problem greatly. Even for a problem with only three locations, the planning horizon greatly affects the size. Figure 3.6 illustrates a graph for a planning horizon of 12 hours.

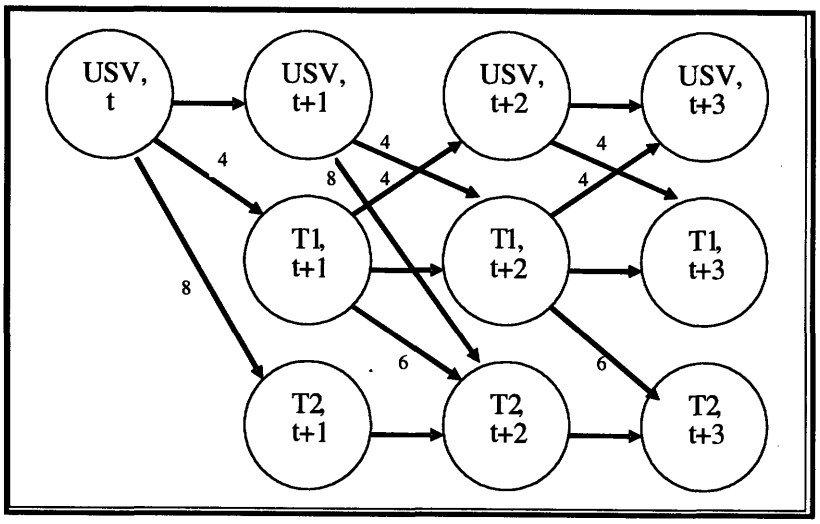



**Figure 3.6: Time-Space Graph Larger Example**

## 3.3 Problem Classification

Now that we have a visualization of our problem, we compare it to problems found in the literature. Problems in the literature that solve vehicle flows on arcs and nodes are known commonly as vehicle routing problems. . Many variations and formulations of routing problems exist. Our goal is a utilize a formulation that handles the elements of time, multiple vehicles, multiple vehicle specifications, very large numbers of target locations, and can be solved and re-solved quickly to account for a dynamically changing environment to maximize the benefit received. The purpose of this section is to classify properly the USVOPP problem in research so we know what work has been previously completed and what methods to explore in solving the USVOPP. In addition, this section describes the basis of the mathematical formulation that we use for the USVOPP.

### 3.3.1 Mathematical Programming Background

Vehicle routing problems are omnipresent around us. A field of research dedicated to solving them is known as Linear Programming. Linear programming uses mathematical models to represent real-world problems. It is linear in that the objective and all of the constraints are linear functions of the decision variables. The goal of linear programming is to either maximize or minimize the objective function value, which represents either rewards or costs depending on

the problem formulation. It does this with respect to a set of constraints that represent the conditions that limit the problem.

The conventional linear programming problem requires that the formulation decision variables are continuous and come from the domain of non-negative real numbers. This, however, is not the case for many problems and not for the USVOPP. The USVOPP has decision variables that are constrained to be discrete integers. Specifically, the decision for whether to send an USV to a target or not is a binary decision, either yes or no. We cannot send half an USV to one target and half to another. In addition, we have positive continuous real-value decision variables for the idling time at each location. Formulations that include binary decision variables and continuous real-valued variables are known as Mixed Integer Programming (MIP) problems, and fall under the mathematical programming classification of Integer Programming.

When formulated as a MIP, our USVOPP is a unique type of problem that falls under the more general class of problems known as the Traveling Salesman Problem (TSP). TSPs also use a network of nodes and arcs to represent the problem. However, our problem has significant differences from the standard TSP.

### 3.3.2 Literature Review

In this section, we introduce the literature of some models relevant to the USVOPP. The TSP and its variants are presented, and algorithms for this class of problems are reviewed. In this section, we discuss the problem definitions, formulations, and both exact and heuristic algorithms formulated to solve the various problems. However, these algorithms do not incorporate all of the types of constraints that are critically important in the USVOPP. In addition, generally the algorithms given in the literature were applied to much smaller scale problems than the USVOPP. This section describes the relationships of the different types of formulations to the USVOPP. In the end, we define a classification of the team orienteering problem with time windows as the basis for our model for the USVOPP.

### 3.3.2.1 Traveling Salesman Problem (TSP)

One of the classic problems in Operations Research that mathematical programming is applied to solve is the TSP. In this section, we partly reference the work of Bodin, L., Golden, B., Assad, A., and Ball, M. [13] to explain the TSP.

#### 3.3.2.1.1 Problem Definition

The classical Traveling Salesman Problem is frequently referred to as the TSP [50]. A traveling salesperson wants to visit each node of a set of nodes, or cities for example, exactly once, starting from, and returning to his original starting point, or home. The objective is to minimize the total distance traveled by the salesperson.

Mathematically, the TSP problem can be defined as follows:

Consider a graph (N, A), where N = {1, 2,..., n} is a set of vertices, containing n vertices which represent cities to visit, and A = {(i,j): i, j ∈ N} is an arc set. Associated with each arc, (i, j), is a nonnegative cost, denoted by $c_{ij}$. The TSP is this:

Given (N, A) and C = { $c_{ij}$ }, find an optimal route from and back to the starting vertex, covering every vertex in the network exactly once, with the least total cost. When $c_{ij} = c_{ji}$ for all $i$, $j \in 1,\ldots, n$, the problem is symmetrical; when $c_{ij}$ does not necessarily equal $c_{ji}$, the problem is asymmetrical. We consider only the asymmetrical one because the symmetrical problem can be treated as a special case of the asymmetrical one.

#### 3.3.2.1.2 TSP Algorithms

To solve these problems when the number of nodes becomes large is very difficult. The algorithms to solve the TSP can be classified into two categories: exact algorithms and approximate algorithms. The known exact algorithms that search the whole solution space can be as difficult as exhaustive search (a particular exact algorithm) in the worst case. Known approximate algorithms avoid searching the whole solution space and attempt only to find a solution that is within certain proximity of an optimal solution.

Exact algorithms:

It is a mathematical conjecture that the complexity class of the TSP is Nondeterministic Polynomial-time Complete or *NP*-Complete. A problem is in class NP if we can check in polynomial time whether a "lucky guess" is actually a solution. For a given directed graph with edge costs, the problem of determining whether there is a path of cost C (for any particular C) is in NP – if somebody gives us a path, we can check in polynomial time whether the cost of that path is C. Any particular TSP can be solved by solving a sequence of these "does there exist a path of cost C" special problems; we choose the C's as a binary search sequence. A problem is NP-complete if whenever we have an engine that solves this problem in polynomial time, you can use this engine as part of an algorithm that solves any other problem in class NP in polynomial time. This means that we cannot find a solution algorithm that gives an optimal solution in a time that has polynomial variation with the number of nodes (n), as in $n^x$. The best we can currently do is to solve the problem that varies exponentially with the number of nodes, as in $2^n$. This makes solving TSP problems intractable for a large number of nodes. Because the TSP is a *NP*-Complete problem, it is very difficult to solve optimally. For example, a complete graph with N vertices requires (N-1)! ways to choose a circuit of length N. As N becomes large, the number of possible sequences explodes.

Algorithms do exist that are usually good at solving the TSP exactly. The branch and bound algorithm is used frequently to solve the TSP exactly. However, in the worst case, branch and bound is equal to exhaustive search. The origin of the branch and bound algorithm goes back to the work of Dantzig, Fulkerson, and Johnson [21]. The branch and bound method solves a discrete optimization problem by breaking up its feasible set into successively smaller subsets, calculating bounds on the objective function value over each subset, and using them to discard certain subsets from further consideration. The bounds are obtained by replacing the problem over a given subset with an easier (relaxed) problem, such that the solution value of the latter bounds that of the former. The algorithm ends when either each subset has produced a feasible solution or has been shown to contain no better solution than the one already attained by the algorithm. The best solution found during the algorithm is a global optimum.

Other exact algorithms such as cutting planes and dynamic programming have also been used to find a global optimal solution to the TSP. Exact algorithms which involve exhaustive search of all possibilities are limited to problems with a relatively small number of vertices and,

usually, with only a few constraints. Even when they are applicable, exact algorithms tend to require more computing time compared to approximate algorithms, which are described next.

Approximate Algorithms or Heuristic Algorithms:

Laporte [39] classified heuristic algorithms into two categories: tour construction procedures, which incorporate vertices systematically into a solution, and tour improvement procedures, which first generate a feasible, but not optimal, solution and then improve the solution by repeatedly removing and adding vertices into the solution. Composite procedures construct a starting tour from one of the tour construction procedures and then attempt to find a better tour using one or more of the tour improvement procedures.

Tour Construction Procedures:

We explain one of the frequently used methods, the Nearest Neighbor Procedure by Rosenkrantz and Lewis [52].

Step 1: Start with any vertex or node as the beginning of a path.

Step 2: Find the vertex closest to the last vertex added to the path. Add this vertex to the path.

Step 3: Repeat step 2 until all vertices are contained in the path. Then, join the first and last vertices.

This procedure requires approximately $n^2$ computations. There are other tour construction procedures that fall under the classification of Insertion Procedures; examples include Nearest Insertion, Cheapest Insertion, Arbitrary Insertion, Greatest Angle Insertion, and others [52].

Tour improvement procedures:

Perhaps, the best-known heuristics for the TSP are the branch exchange heuristics introduced by Lin [41]. These heuristics work as follows:

Step 1: Find an initial tour. Generally, this tour is chosen randomly (but not required) from the set of all possible tours.

Step 2: Improve the tour using one of the branch exchange heuristics.

Step 3: Continue step 2 until no additional improvement can be made.

In the general sense, a certain number of edges, k, in a feasible tour, are exchanged for k edges that are not in the solution as long as the result remains a tour and the length of that tour is less than the length of the previous tour. Exchange procedures are referred to as k-opt procedures where k is the number of edges exchanged at each iteration. Normally, 2-opt or 3-opt

are the most common procedures in practice. They are used to generate excellent solutions to large-scale TSPs in a reasonable amount of time.

Composite procedures:

The basic composite procedure can be stated as follows:

Step 1: Obtain an initial tour using one of the tour construction procedures.

Step 2: Apply a 2-opt procedure to the tour found in step 1.

Step 3: Apply a 3-opt procedure to the tour found in step 2.

The composite procedure is relatively fast computationally and gives excellent results. The idea behind the composite procedure is to get a good initial solution rapidly and hope that the 2-opt and 3-opt procedures will then find an almost-optimal solution. In this way, the 3-opt procedure, which is the computationally most expensive step of the three, need only be used once [13]. Laporte [39] noted that the best approach to use would be composite procedures, which combine both the tour construction and tour improvement procedures. The TSP is a good starting point, but because the USVOPP has time window requirements on each vertex, we need to explore methods that handle the extra complexity.

#### 3.3.2.2 The Traveling Salesman Problem with Time Windows (TSPTW)

The TSPTW is defined as follows:

A single traveling salesman is required to visit a set of N vertices on a tour constructed on a fully connected network, just like the original TSP. He begins at a designated vertex and returns to the same vertex at the end of the tour. There is a cost associated with traveling between any vertex pair. In addition, each vertex has a time window in which it must be visited. The objective is to visit all vertices exactly once within their time windows at minimum cost.

The exact algorithm to solve the TSPTW was introduced by Mingozzi et al. [44]. They utilized dynamic programming strategies for the TSPTW and considered precedence constraints as well. Precedence constraints ensure that the route constructed is in time order. For example, the first vertex is visited at time 2, and the second vertex is visited no earlier than the preceding vertex. In 1998, Gendreau et al. [27] mentioned a heuristic method, a generalized insertion heuristic for the TSPTW. This formulation is closer to what the USVOPP requires, but since our

goal is to maximize the value collected at each point, and not necessarily visit each point and find the minimum distance, we need to explore a method that maximizes value.

### 3.3.2.3 The Prize Collecting Traveling Salesman Problem (PCTSP)

The prize collecting traveling salesman problem (PCTSP) is the problem where a salesperson travels from one customer to another at a given cost, pays a penalty for every customer he fails to visit and collects a prize for each customer he visits. This type of problem is also known by other names such as the selective TSP, TSP with profits, and others. The salesperson's objective is to collect an amount of prize money greater than or equal to a given amount while minimizing the sum of his travel costs and penalties to be paid [25]. When the given amount g is equal to the total amount of all possible prizes collected from all customers, the PCTSP is reduced to the TSP.

The PCTSP was introduced by Balas and Martin [9] as a model for scheduling the daily operation of a steel rolling mill. Balas [10] discussed the structural properties of its polytope and Fischetti and Toth [25] developed a branch and bound algorithm for its optimal solution. This problem formulation is closer to the desired formulation than the TSPTW, especially if we add time windows. However, we are not concerned with setting a certain amount to collect, and then minimizing distance. In addition, the USVOPP does not have the requirement of starting and finishing in the same location.

### 3.3.2.4 The Orienteering Problem (OP)

The Orienteering Problem (OP) can be described as follows: given n vertices, each vertex i has a score $s_i \geq 0$ and the scores of the starting vertex, denoted by 1, and the ending vertex, denoted by n, are set to 0; i.e., $s_1 = s_n = 0$. The arc between vertices i and j has a cost $c_{ij}$ associated with it. Since n vertices are usually considered in the Euclidean plane, and the distance and travel times between vertices are determined by the geographical measure, they are assumed to be known quantities and distance is used as the representative of cost. Each vertex can be visited at most once. Therefore, the objective of the OP is to maximize the score of a

route that consists of a subset of vertices starting from vertex 1 and finishing at vertex n without violating the cost (distance) constraint max T. Another common name for the OP is the Generalized TSP.

While the OP was originally modeled and applied for the sport of orienteering, it has practical applications in production scheduling and vehicle routing as discussed by Golden et al. [31]. They also proved that the OP is NP-hard. This is worse than NP-complete, but has NP-complete problems as special cases. It should be noted that the OP with very close starting and ending vertices can be transformed to the PCTSP. They solve the PCTSP with all possible values of g. The maximum value of g that results in a feasible solution for the PCTSP is the result of the corresponding OP. Meanwhile, The PCTSP can be reduced to the TSP when g is equal to the total prize of all vertices. In some situations, when the starting and ending vertices of the OP are very close and the time constraint T is great enough to cover all vertices, the OP is equivalent to the TSP. Solving the TSP in that case results in a feasible solution of the OP. Work has been done on exact methods for the OP such as integer programming, dynamic programming, and branch-and-cut algorithms. Although these approaches have yielded solutions to smaller sized problems, as in other NP-hard problems, the computational limitations of exact algorithms encourage the exploration of heuristic procedures.

#### 3.3.2.4.1 Algorithms for the OP

Exact Algorithms:

The branch-and-cut algorithm of Fischetti et al. [26] could be used to solve the OP exactly. It consists of a two-phase method. In the first phase, the algorithm avoids branching by adding branch cover cuts. In the second phase, the algorithm works on a sparse graph (resulting from the branch cover cuts produced in the first phase), and utilizes a classical branching strategy to close the integrality gap and converge to the solution. Other exact methods have also been developed to solve the OP, such as dynamic algorithms and other linear relaxation algorithms.

Heuristic Algorithms:

The first heuristics for the OP, the S-algorithm and the D-algorithm, were proposed by Tsiligirides [62]. The S-algorithm uses the Monte Carlo Simulation method to construct routes using probabilities correlated to the ratio of vertex score to vertex distance from the current

vertex. The D-algorithm is based on the vehicle scheduling method proposed by Wren and Holiday [67]. This approach operates by dividing the search area into sectors that are determined by two concentric circles and an arc of the known length. Sectors are varied by changing the two radii of the circles and by rotating the arcs. A route is built when all vertices in a particular sector have been visited, or it is impossible to visit any other vertex of the same circle without violating the max T constraint. Tsiligirides [62] also proposed the most well known test problems for the OP, which have 21, 32, and 43 vertices. Golden, Levy and Vohra [31] proposed an iterative heuristic for the OP, which consists of three steps: route construction using a greedy method, route improvement that uses a 2-opt swap, and center-of-gravity which guides the next search step. Golden, Wang and Liu [32] combined Tsiligirides's S-algorithm concept (randomness), the center of gravity, and learning capabilities into another approach to solve the OP. To provide probabilities for vertex selection, the score of neighboring vertices are also considered. Wang, et al. [64] proposed an artificial neural network approach to solve the OP. A Hopfield-like neural network is formulated and a fourth order convex energy function is devised. Ramesh and Brown [47] proposed a four-phase heuristic algorithm for the generalized orienteering problem, i.e., the cost function is not limited to a Euclidean function. The four phases consist of vertex insertion, cost improvement, vertex deletion, and maximal insertions. The route is improved by a 2-opt procedure followed by a 3-opt procedure in the second phase. In the third phase, one vertex is removed from the current route and one vertex is then inserted in an attempt to decrease the length of the route. Finally, as many unassigned vertices as possible are inserted onto the current route in order to increase the total score. Chao, et al. [16] introduced a two-step heuristic to solve the OP. In the first step, initialization, by using the starting and ending vertices as the two foci of an ellipse and the max T constraint as the length of the major axis, several routes are generated and the one with the highest score is the initial solution. The initial route is then improved by a 2-vertex exchange in the cheapest-cost way, and then improved by a 1-vertex improvement that tries to increase the total score. They applied this algorithm to Tsiligirides's problems [62] and 40 new test problems. This framework is much closer to our envisioned USVOPP formulation, but we still have not included time windows for the OP and have not considered multiple vehicles.

#### 3.3.2.5 The Team Orienteering Problem (TOP)

The Team Orienteering Problem (TOP) is an extension to the OP. The distinction being that the TOP has the generalization to the case of multiple tours of the OP. In relation to the sport of orienteering, it involves a team of contestants trying to coordinate together to collect as many points as possible in a given amount of time. In the multiple vehicle case, this relates to the classic vehicle routing problem classification. Other names for the TOP include the multiple depot vehicle routing problem with profits, multiple depot selective vehicle routing problem, or multiple tour maximum collection problem.

Only very recently have researchers begun to explore the TOP. Because of the difficulty of the problems mentioned prior to this, most of the research done today in this field is focused on the multiple vehicle TSPTW. The focus is to find better and faster methods to solve larger and more complex problems. Some research has been done on this problem and a few heuristics have been developed for the TOP. In 2004, Hao Tang and Elise Miller-Hooks developed a TABU search heuristic for the TOP [56]. In 2005, C. Archetti, A. Hertz, and M.G. Speranza compared TABU search and Variable Neighborhood search and concluded that Variable Neighborhood search outperformed two different TABU search heuristics [8]. This formulation is very close to what we want to model the USVOPP to; only without the time windows the USVOPP requires.

#### 3.3.2.6 The Team Orienteering Problem with Time Windows (TOPTW)

The closest formulation for our model is the Team Orienteering Problem with Time Windows. To the best of our knowledge, no formal papers have addressed this exact problem to date. With the addition of time windows to the TOP, the USVOPP is very similar to the TOPTW framework. Figure 3.7 illustrates an example solution to a USVOPP with two USVs each visiting four targets in the planning horizon.

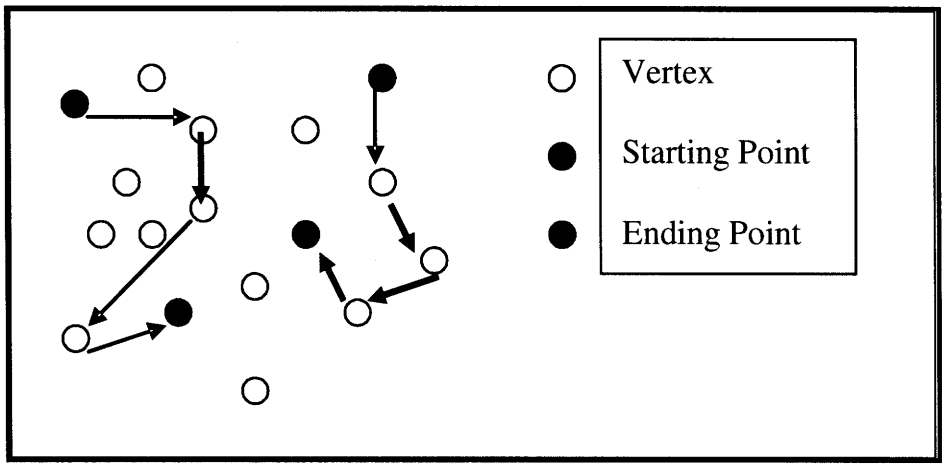



**Figure 3.7: Orienteering Problem Solution**

### 3.3.3 Relation of TOPTW to the USVOPP

To the best of our knowledge and assumptions, a modified TOPTW can be applied to solve the USVOPP. There are a few minor differences between the TOPTW and the USVOPP. The TOPTW requires the vehicles to return to the home location. The USVOPP is open ended. In order to take advantage of this model we map the real-world characteristics of the USVOPP to the objective and constraints of an open-ended TOPTW. A specific target is worth some scientific value depending on the location and point in time. Therefore, we construct an objective function to reflect these values by prizes in the USVOPP formulation model. We will explain the notations and assumptions in detail in Chapter 4 and Chapter 5 when we develop and test the models for the USVOPP.

## 3.4 Choosing the Solution Method

The choice of solution method for solving the USVOPP problem depends on solution techniques considerations (such as existing algorithms in the available literature), available computer software, computer/hardware, and time. In order to limit the search for existing algorithms to solve the problem, it is first considered whether to use exact or heuristic algorithms.

When solving an optimization problem, we are really only optimizing a model of a problem originating in the real world. There is no guarantee that the best solution to the model is also the best solution to the underlying real-world problem. Two reasons for why the best solutions do not match are that some real-world constraints can be decided to be omitted in the model, because they are not considered important, and the numbers used in the model/implementation are not precise (e.g. estimated measurements). Even though heuristics are not guaranteed to provide us with an optimal solution to the underlying problem, neither are exact methods. Furthermore, heuristic methods are usually more flexible and capable of coping with more complicated and realistic objective functions and/or constraints than exact algorithms. Another reason for considering heuristics for solving the USVOPP is the complexity of the USVOPP, which makes the problem solvable using exact algorithms only for small problem sizes.

We will cover the different available heuristics and compare them in detail in the next chapter. After we present the formulation of the problem, we analyze which heuristics could potentially solve the USVOPP. We then adapt our heuristics of choice from the given heuristics to solve the USVOPP to meet our requirements for speed, efficiency, and additional constraints. Because no research has directly been applied to the TOPTW, we should compare a couple of different heuristics to compare how our heuristic performs.

**[This Page Intentionally Left Blank]**

# Chapter 4

# Problem Formulation

In this chapter, we propose a mathematical formulation for the USVOPP, which is adapted from Orienteering Problem models in the literature. The first section explains what data is required and how we converted the real-world scenarios into a mathematical representation useful for our tests. The second section presents the mathematical notation and MIP formulation of the USVOPP. The third section presents a literature review of the various heuristics that could solve the USVOPP. The fourth section describes the various heuristics we developed and implemented to solve the theoretical USVOPP.

## 4.1 Data Representation

We envision a USVOPP scenario consisting of the following data:

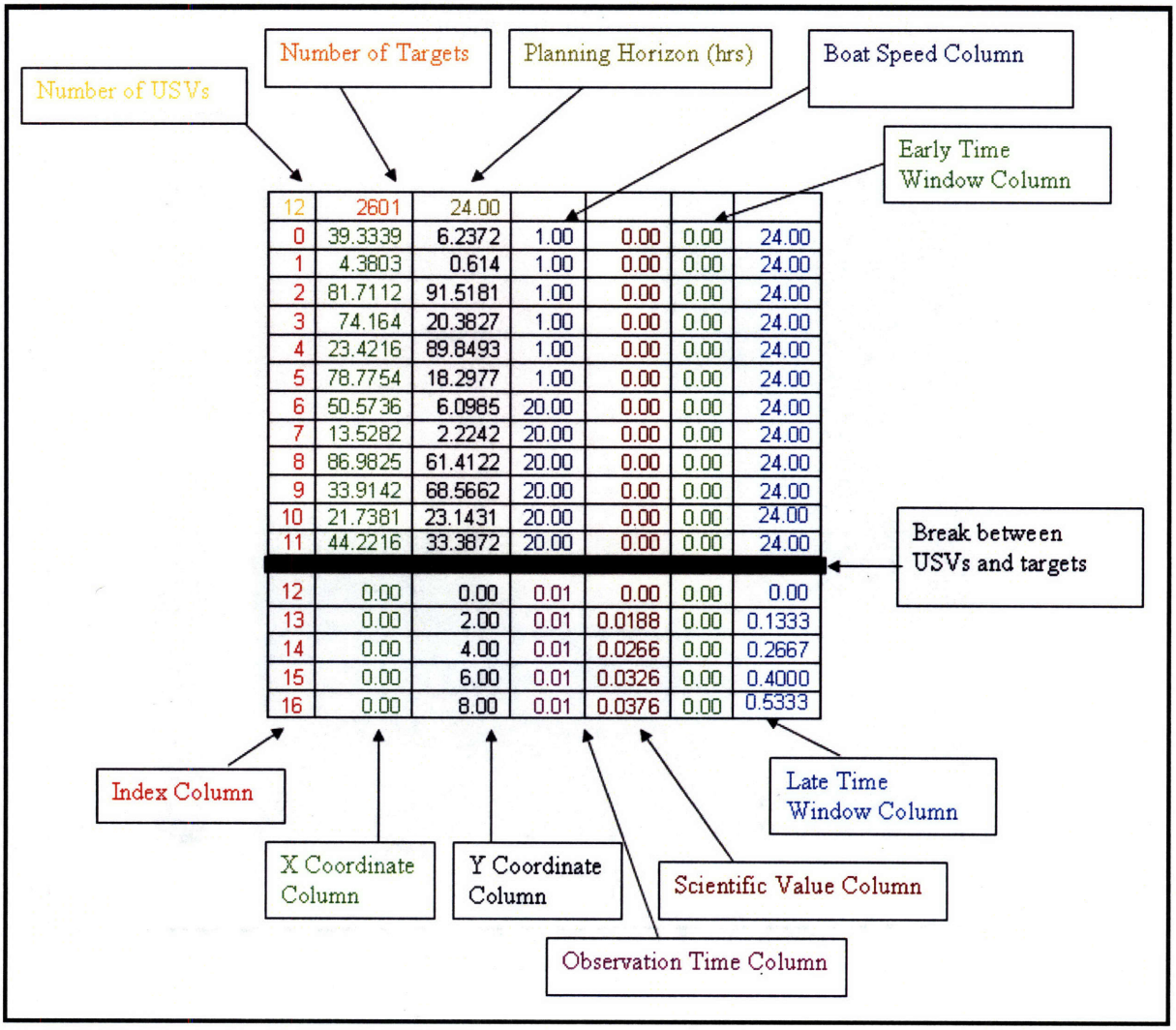
1. Initial locations of all USVs
2. Locations of all targets
3. Speed of each USV
4. Time Windows of Opportunity for each target
5. Planning horizon
6. Required Observation Time
7. Target Observation Value

The geographical locations of the USVs and targets are represented by latitude/longitude coordinates. Without loss of generality, we represent the initial USV and target locations with x-y coordinates as opposed to latitudes and longitudes. Each target has an associated time window of opportunity represented with an early and late time, a required observation time to observe the target, and a value that represents the scientific value added if observed. To explain the data in detail, we examine a sample dataset.

### 4.1.1 Sample Data

Table 4.1 illustrates the first 17 rows of a dataset with 2613 rows for an example hurricane scenario. The top row of the data represents the initialization data for the number of USVs, number of targets, and the planning time horizon. The sample data given initializes with 12 USVs, 2601 targets, and a planning horizon of 24 hours respectively reading from left to right across the first row. The next row of data represents the data for the first USV. The USV data is always initialized first. The number of USVs initialized in the first row of the data gives the computer the number of rows that represent USV data. For this scenario, the next 12 rows of data represent the USV data because the first number in the first row is 12. For each USV data row, the first column holds an index number, the second holds the x coordinate location, the third holds the y coordinate location, the fourth holds the USV speed, and the remaining columns are ignored. For each target data row, the first column holds an index number, the second holds the x coordinate location, the third holds the y coordinate location, the fourth holds the required observation time, the fifth holds the scientific value, the sixth holds the early time window value, and the seventh holds the late time window value. The datasets are generated in this format because this is similar to how J.-F. Cordeau and M. M. Solomon [20] set up standard multi-depot

vehicle routing problems with time window datasets. A sample table of the data follows in Table 4.1.




| | | | | | | |
|---|---|---|---|---|---|---|
| 12 | 2601 | 24.00 | | | | |
| 0 | 39.3339 | 6.2372 | 1.00 | 0.00 | 0.00 | 24.00 |
| 1 | 4.3803 | 0.614 | 1.00 | 0.00 | 0.00 | 24.00 |
| 2 | 81.7112 | 91.5181 | 1.00 | 0.00 | 0.00 | 24.00 |
| 3 | 74.164 | 20.3827 | 1.00 | 0.00 | 0.00 | 24.00 |
| 4 | 23.4216 | 89.8493 | 1.00 | 0.00 | 0.00 | 24.00 |
| 5 | 78.7754 | 18.2977 | 1.00 | 0.00 | 0.00 | 24.00 |
| 6 | 50.5736 | 6.0985 | 20.00 | 0.00 | 0.00 | 24.00 |
| 7 | 13.5282 | 2.2242 | 20.00 | 0.00 | 0.00 | 24.00 |
| 8 | 86.9825 | 61.4122 | 20.00 | 0.00 | 0.00 | 24.00 |
| 9 | 33.9142 | 68.5662 | 20.00 | 0.00 | 0.00 | 24.00 |
| 10 | 21.7381 | 23.1431 | 20.00 | 0.00 | 0.00 | 24.00 |
| 11 | 44.2216 | 33.3872 | 20.00 | 0.00 | 0.00 | 24.00 |
| 12 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 |
| 13 | 0.00 | 2.00 | 0.01 | 0.0188 | 0.00 | 0.1333 |
| 14 | 0.00 | 4.00 | 0.01 | 0.0266 | 0.00 | 0.2667 |
| 15 | 0.00 | 6.00 | 0.01 | 0.0326 | 0.00 | 0.4000 |
| 16 | 0.00 | 8.00 | 0.01 | 0.0376 | 0.00 | 0.5333 |

**Table 4.1: Sample Data**

Some of the data represented in Table 4.1 with the addition of the rest of the 2595 targets is illustrated more intuitively with a graph. Figure 4.1 illustrates a sample hurricane scenario where the eye of the hurricane is in the bottom-left corner of the graph. Each colored square in the figure corresponds to a target specified in Table 4.1. The white dots in the figure represent the USV starting locations. The scale on the right side of the figure illustrates the relative scientific value of each point in the area. The number corresponding to each scientific value is externally specified, and how that occurs is not discussed further in this thesis. The dark red on

the graph represents an area with high importance (high uncertainty or high benefit for prediction model) and the blue represents an area with lower importance (low uncertainty or low benefit for prediction model) for the next planning period. The time windows, observation times, vehicle speeds, and planning horizon are not illustrated in this graph.

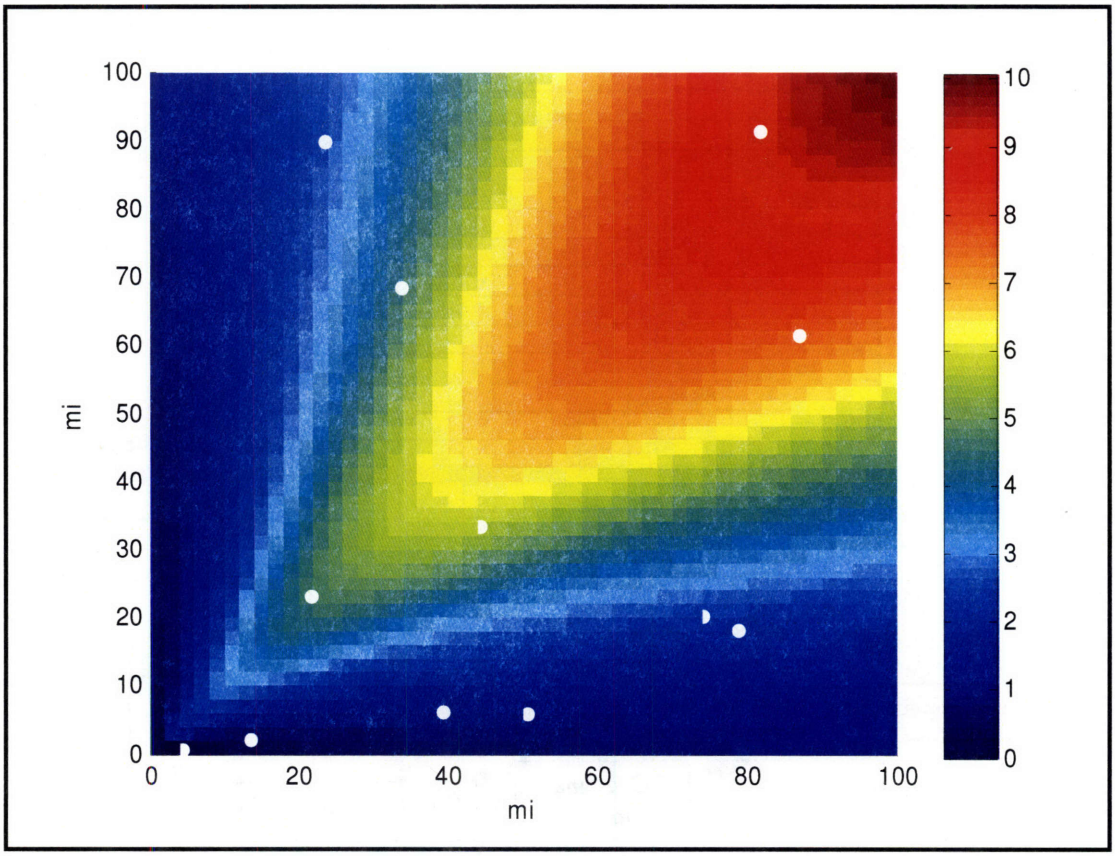



**Figure 4.1: Sample Hurricane Graph**

### 4.1.2 Mathematical Estimation of HAB and Hurricane Scenario Data

In this section, we describe how we create test data for the HAB and hurricane scenarios. Because the USVs have not yet been implemented, and their purpose is to provide the means necessary to collect HAB data, we do not have real algal bloom data. Data for past hurricane paths and intensities exist, but for the purposes of testing our algorithm we chose to simplify the data and create simple mathematical hurricane paths because we desire generic data that is easy to create and test. To create the datasets, we used MATLAB software to construct the data and plot the graphs of the data.

#### 4.1.2.1 Hurricane Scenario Data

Many different models exist today that attempt to predict hurricane movement, intensity, and surge. Both statistical and dynamic models exist. Statistical models predict hurricane behavior by considering the historical behavior of "similar" hurricanes. Dynamic models predict hurricane behavior by the use of thermodynamic and fluid mechanics models; the partial differential equations given by these models are solved numerically, with initial conditions that correspond to physical observations of wind velocity, pressure, temperature, and relative humidity. Generally, it is best to have a large number of physical observations.

The data collected by the USVs is meant to support a dynamic model's prediction. The addition of the sub-surface water temperature could add another element to a dynamic model, which could require this data. We assume that observing the underwater temperature with the USVs is valuable to the dynamic hurricane-forecasting model. For our test scenarios, we assume that a hurricane has a predicted straight-line path from the eye of the hurricane to the end of the desired observation area from the latest dynamic model output available. The values we have specified in Figure 4.1 are artificial, but are meant to suggest that observations along the projected path of the hurricane are more important than observations far from this path. In addition, targets further from the current position of the hurricane are more important than targets near the current position.

We estimate the scientific value at each point in the grid with a simple mathematical function. For each x-y coordinate pair, starting with (0,0), the value function is the radial distance away from the eye, $r = \sqrt{x^2 + y^2}$, raised to a smoothing power value of choice $(s)$. For the example data in Table 4.1, we used 1.5 and multiplied r by the normal probability distribution function evaluated at the absolute value difference of x and y, with mean $\mu = 0$, and variance $\sigma^2 = \frac{r}{2}$. We assume that the uncertainty in the grid can be adequately estimated with a normal distribution function. This creates a bell shaped fit with the peak of the bell on the trajectory of the hurricane. The variance that we use for the normal probability function is the radial distance away from the eye of the hurricane divided by 2 so that the variance fits the grid and points on the edge of the grid have minimal scientific gain. In equation form:

$$V = r^{0.5} * n(|x-y|, \mu = 0, \sigma^2 = \frac{r}{2}) \text{ where } n(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.1)$$

This function creates a scientific value field that is adjustable to any defined parameters for the normal probability distribution function, or storm size, or path.

The time windows for the scenario are dependent on the estimated speed of the hurricane. We assume the early time window for all locations is zero because we assume that at the beginning of the scenario, the value is determined to be important right now. The late time window is proportional to the speed of the hurricane and the distance away from the eye. The points closest to the hurricane have the earliest late time window values and the points furthest away have the latest late time window values. For example, hurricanes travel, on average, 10 to 20 knots across the water. For a point 30 nautical miles away from the eye of the storm, assuming the hurricane is traveling at 15 knots, the upper time window is 2 hours.

#### 4.1.2.2 HAB Scenario Data

For the HAB scenario, the construction method is more complicated. We know that the Woods Hole Institute desires to, "Develop a regional, rapid-response communications network to facilitate coordinated sampling and interpretation of HAB events and provide early warning and data dissemination," [24]. We assume that the Prototype Phytoplankton Fluorescence Sensing System Fluorometer on a USV can collect data to help the network's predictions. The *Global Ecology and Oceanography of Harmful Algal Blooms* reported, "Despite the proven utility of models in so many oceanographic disciplines, there are no predictive models of population development, transport, and toxin accumulation for any of the major harmful algal species," [1]. Therefore, there is a clear need to develop realistic physical models to monitor and predict HABs. Because we do not have data or information on the spread of HABs, we create plausible scenarios to test our algorithms for the USVs. HABs can last days, weeks, or months and move typically very slowly. The USVs could be the tool that will enable future models to measure growth rates and predict the spread of algal blooms.

For our HAB scenario, we assume that the HAB has been identified, and is spreading over time. We assume that the HAB spreads in an elliptical manner and typically only .02-.05 knots in addition to the tidal current. We assume that the boats observe the HAB as it spreads.

Figure 4.2 illustrates four snapshots of a notional example HAB spreading over four days.

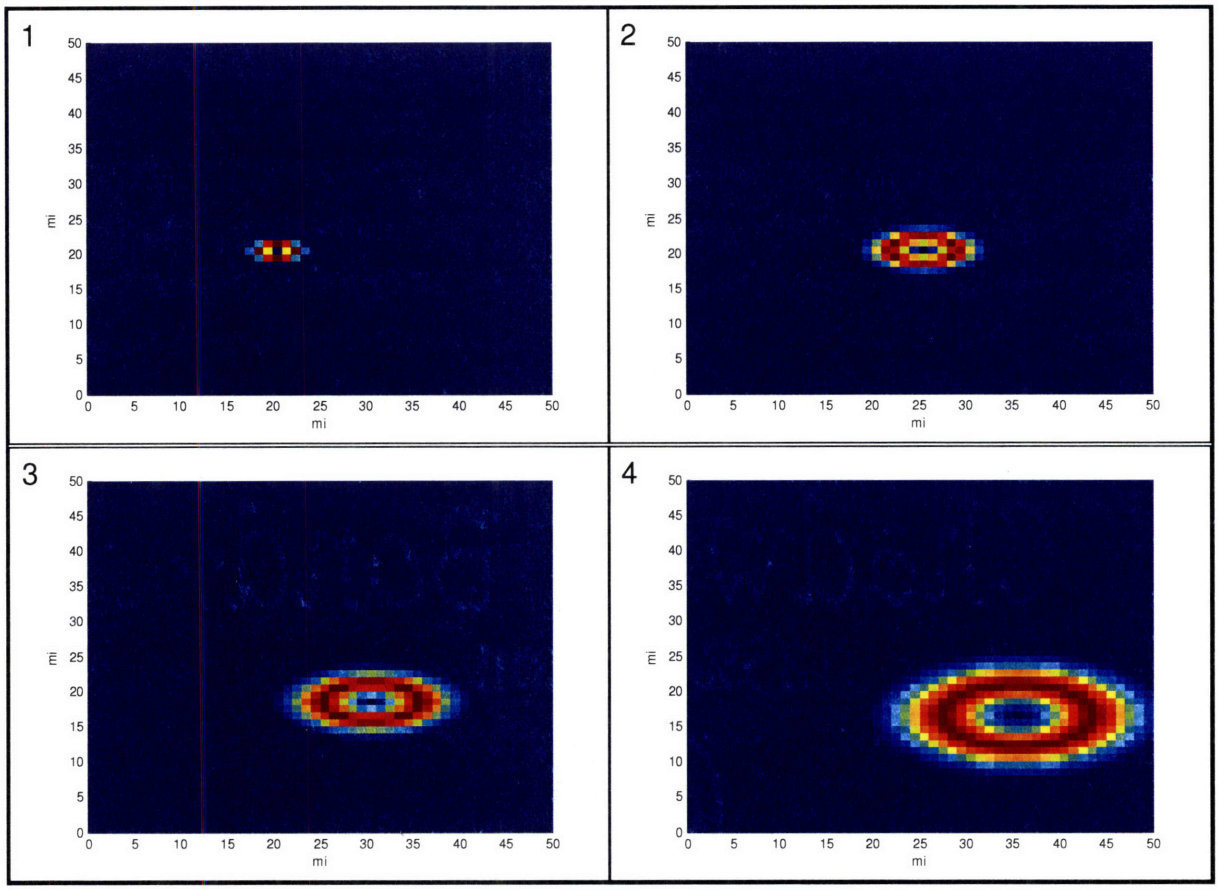



**Figure 4.2: Sample HAB Spread**

We estimate the scientific value at each target in the grid with a simple mathematical function. For each x-y coordinate pair, starting with (0,0), the value function is the beta probability density function evaluated at the elliptical distance away from the center of the bloom (xc,yc), $r = \frac{(x-xc)^2}{a^2} + \frac{(y-yc)^2}{b^2}$, where a and b are the semi-major axis and semi-minor axis assuming that a>b, with mean $\mu = 0$, and variance $\sigma^2 = \frac{r}{2}$. We assume that the uncertainty in the grid can

be adequately estimated with a beta distribution function with the two free parameters, $\alpha$ and $\beta$ set to 2 and 15 respectively. This creates a desired shaped fit with the peak on the expected ellipse of the HAB. Figure 4.3 illustrates the shape of the desired probability density function that represents the value of the data observed in a straight-line from the center of the ellipse at zero to the end of the grid. In equation form:

$$V = \mathrm{B}(r, \alpha = 2, \beta = 15) \text{ where } \mathrm{B}(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,dt} \tag{4.2}$$

This function creates a value field that is adjustable to any defined parameters for the beta probability distribution function to represent the values for any HAB size or path. As a HAB spreads, the same points could be observed again, but will have different scientific value.

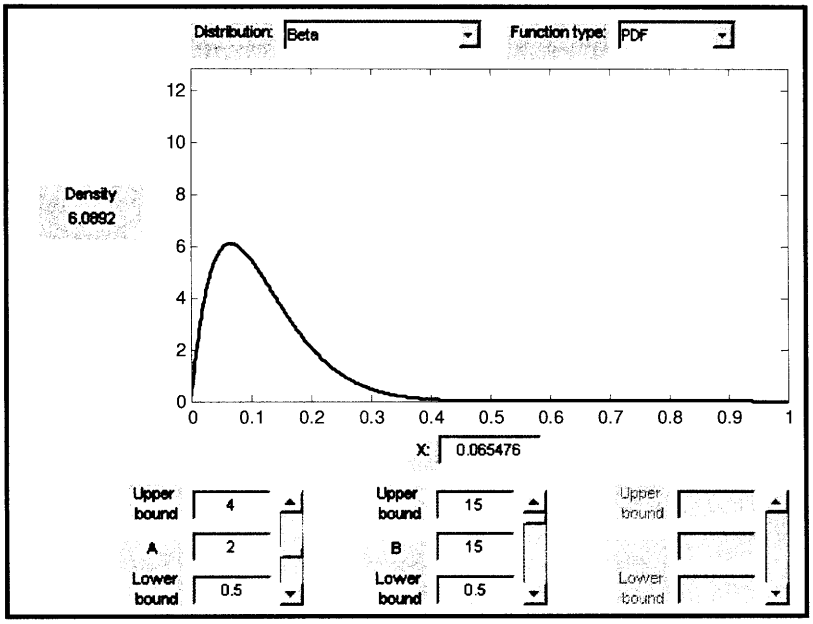



**Figure 4.3: Beta Function PDF with A=2, B=15**

The value of observing any particular location is the value of a certain beta probability density function. Unfortunately, this beta probability density function changes as time passes, in a way that models the growth of the HAB. Therefore, the value of observing a particular target depends on the time of the observation. The input format we use to describe a scenario does not allow values that are time varying. We only specify one number for value. The input format

does allow us to specify time window, though. Therefore, we can select the time window at a given location to be the interval during which the value (computed using the beta probability density function) is larger than a particular threshold.

### 4.1.3 Translating Data to Mathematical Programming Formulation

With the ability to create notional datasets, we can convert the data and requirements from the created scenarios into a mathematical presentation. In the following section, we describe how to convert the scenario data of USV and target locations and routes, into a graph of targets and arcs that conform to formal mathematical programming formulation terminology.

**USV Locations:** Each initial USV location is represented by a vertex of type $u$ of the graph. USVs that have the same initial location are represented by unique nodes. We write $U$ to denote the set of all USV Locations.

**Target Locations:** Each target location is represented by a vertex of type $l$ of the graph. Target locations with the same coordinates, but different time windows, are represented by unique nodes. In addition, targets and initial USV locations at the same location have unique nodes. We write $L$ to denote the set of all target locations.

**Routes:** The route between an initial target location and a target or two target locations is represented by the arc connecting two corresponding vertices in the graph. Vertices between USV initial locations are not necessary because no value is added for traversing these arcs. We write $A$ to denote the set of all arcs.

**Traveling Time:** The time a USV spends when traveling from one location to another is represented by the time along the arc connecting two corresponding vertices in the graph. We compute the travel times by using a simple technique that assumes that USVs travel at a constant speed, and that it is feasible to travel in a straight line between any two targets and that tidal currents can be ignored. The set of all "time arcs" is denoted by the matrix *travelTime*.

**Observation Time:** The time a USV spends serving a location is represented by the time of serving the corresponding vertex in the graph. The set of all observation times at vertices is denoted by *observation*.

**Idle Time:** The time an USV spends idling at a location is represented by the idle time at the corresponding vertex in the graph. We use two decision variables, *arrival* and *departure*, to

represent the arrival time at a target and the departure time at a target. The set of all idle times at each vertex is inferred in the model by calculating the difference between the departure time and the arrival plus observation times at each target for each vertex visited.

$$idletime_i = departure_i - (arrive_i + observation_i) \quad (4.3)$$

If the vertex was not visited, then the equation is irrelevant and the idle time is zero.

**Time Windows**: The time window of opportunity of a target is represented by the time window of the corresponding vertex. For a USV location, there is no time window. In case of a long planning horizon, there could be a set of multiple time windows on each vertex. The sets of all the earliest and latest observation times are denoted by *early* and *late* respectively.

**Value**: When observing a location, USV's collect value for the mission. The value is represented by the score of the corresponding vertex. The set of all scores at vertices is donated by *value* .

**Time**: The planning horizon is notated by T.

**Objective**: The objective of finding an optimal trip is converted to the problem of finding a path from the starting vertices in a 2-weighted, undirected, complete graph $G = (M \cup N, A, travelTime, observation, early, late, value)$ that maximizes the total value collected from the observed targets. The value collected must be completed within their time windows, without violating the constraint of the total route time is less than or equal to T for each USV.

## 4.2 Mathematical Model

In this section, we present the USVOPP as a modified TOPTW. We present a linear programming formulation that can readily be formatted to optimization software. We discuss the mathematical notation, the formulation, the objective function, and the constraints of the formulation.

### 4.2.1 Notation

**Sets**

| | | |
|---|---|---|
| $L$ | Set of target locations | In our work, $L$ can have thousands of elements |
| $U$ | Set of initial USV locations | In our work, $U$ can have at most 24 elements |
| $A$ | Set of travel arcs | From each initial USV location to each target location, to and from all target locations |

**Decision Variables**

| | |
|---|---|
| $observe_{ikt}$ | Binary, 1 if target $i$ is observed by USV $k$, at position $t$, 0 otherwise, the position $t$ represents the $t^{th}$ target observed, for example, if USV #2 traveled from its initial location to target 10, $observe_{10,2,1} = 1$, note $t$ has nothing to do with time, only order for all targets, all USVs, and all positions |
| $travel_{ijk}$ | Binary, 1 if arc $(i, j)$ is traveled by USV $k$, 0 otherwise, for all locations and USVs |
| $arrive_{ik}$ | Positive real number, arrival time for USV $k$ at target $i$, for all locations and USVs |
| $departure_{ik}$ | Positive real number, departure time for USV $k$ at target $i$, for all locations and USVs |

**Constants**

| | |
|---|---|
| $observation_i$ | Positive real number, time required for USV to collect data at target $i$, for all targets |
| $travelTime_{ij}$ | Positive real number, travel time from target $i$ to $j$, $travelTime_{ij}$ can be different from $travelTime_{ji}$ |
| $early_i$ | Positive real number, earliest time we accumulate value by observing target $i$ |
| $late_i$ | Positive real number, latest time target $i$ can be observed |
| $horizon$ | Positive real number, length of time of the planning interval |
| $value_i$ | Positive real number, value at target $i$ |
| $m$ | Integer, number of USVs |
| $n$ | Integer, number of targets |

### 4.2.2 MIP Formulation

The resulting mathematical model then becomes:

#### 4.2.2.1 Objective Function

**Maximize**

$$\sum_{k \in U} \sum_{i=m+1}^{m+n} \sum_{t=1}^{n} value_i * observe_{ikt} \tag{4.4}$$

The objective function, equation 4.3, of the USVOPP consists of the total sum of the value received by observed targets. It represents the maximization of the total scientific value over all of the $m$ USVs.

#### 4.2.2.2 Constraints

**Subject to**

$$\sum_{i=m+1}^{m+n} observe_{ikt} \leq 1 \qquad t = 0, n, \forall k \in U, \tag{4.5}$$

- Every position on the route for a given USV $k$, has at most one target assigned.

$$\sum_{k=1}^{m} \sum_{t=0}^{n} observe_{ikt} \leq 1 \qquad i \in L, \tag{4.6}$$

- Every target is assigned to at most one target over each USV route.

$$\sum_{i=1}^{m+n} observe_{ikt+1} - \sum_{i=1}^{m+n} observe_{ikt} \leq 0 \qquad t = 0, n-1, \forall k \in U, \tag{4.7}$$

- Ensures the targets are assigned to the positions in the route in a successive manner. For example, if a target is assigned to position $t$ for a USV's route, where $t$>0, then a target must be assigned to position $t$-1.

$$observe_{jk1} - travel_{kjk} = 0 \qquad j = m+1, m+n, \forall k \in U, \qquad (4.8)$$

- Ensures that each USV's route starts at the corresponding USV's starting location, or if a target is in the second position on a route, $t$=1, then there must be an arc from the starting location to that target.

$$observe_{ikt} + observe_{jkt+1} - travel_{ijk} \leq 1 \qquad t = 0, n-1, i \in L, j \in L, \forall k \in U, \qquad (4.9)$$

- Ensures that if targets $i$ and $j$ hold to successive positions, $t$ and $t$+1, on an USV route, then the arc $(i, j)$ must exist, which means $travel_{i,j}$=1.

$$\sum_{i \in L} travel_{kik} - \sum_{i \in L} travel_{ikk} = 0 \qquad \forall k \in U, \qquad (4.10)$$

- Forces each of the USV routes to end at its starting location, this is a logical constraint that is implemented to force closed routes. Even though the USVOPP requires open routes, by setting the travel time to return to each USV starting location to zero the end result is the same with the last target visited on the route acting as the end of the route.

$$\sum_{j \in U} travel_{ijk} - \sum_{t=1}^{n} observe_{ikt} = 0 \qquad i = m+1, m+n, \forall k \in U, \qquad (4.11)$$

- If target $i$ is on a route, this constraint forces an arc to emanate from it.

$$\sum_{j \in L} travel_{jik} - \sum_{t=1}^{n} observe_{ikt} = 0 \qquad i = m+1, m+n, \forall k \in U, \qquad (4.12)$$

- If target $i$ is on a route, this constraint forces an arc to enter it.

$$departure_{ik} + travelTime_{ij} - a_{ij}(1 - travel_{ijk}) \leq arrive_{jk} \qquad j = m+1, m+n, \forall i \in L, \forall k \in U, \qquad (4.13)$$

- This constraint ensures that there is at least departure time + travel time separation between the targets on each route (further discussion in section 4.2.2.3).

$$early_i(\sum_{t=0}^{n} observe_{ikt}) \leq arrive_{ik} \qquad i = m+1, m+n, \forall k \in U, \qquad (4.14)$$

- This constraint restricts arrival times for each target to fall after early arrival time allowed.

$$arrive_{ik} + observation_i \leq late_i(\sum_{t=0}^{n} observe_{ikt}) \qquad i = m+1, m+n, \forall k \in U, \qquad (4.15)$$

- This constraint restricts arrival times added to the observation time for each target to fall before the latest time allowed.

$$arrive_{ik} + observation_i \leq departure_{ik} \qquad i = m+1, m+n, \forall k \in U, \qquad (4.16)$$

- This constraint ensures the connection between the arrival time and departure time for each target.

$$arrive_{ik} + observation_i \leq horizon \qquad i = m+1, m+n, \forall k \in U, \qquad (4.17)$$

- This constraint ensures all arrivals and observation times occur before the planning time limit.

$$observe_{ikt} \in \{0,1\} \qquad \forall k \in U, \forall i \in L, \qquad (4.18)$$

- This is a binary constraint for each target observed.

$$travel_{ijk} \in \{0,1\} \qquad \forall (i, j) \in A, \forall k \in U, \qquad (4.19)$$

- This is a binary constraint for each arc traveled.

$$arrive_{ik} \in \Re^+ \qquad i = m+1, m+n, \forall k \in U, \qquad (4.20)$$

- This constraint ensures arrival times have to be a positive real number for each target and USV.

$$departure_{ik} \in \Re^+ \qquad i = m+1, m+n, \forall k \in U, \qquad (4.21)$$

- This constraint ensures departure times have to be a positive real number for each target and USV.

#### 4.2.2.3 Linearization of Precedence Constraints

Equation 4.12 was not the first constraint developed to ensure the consistency of the time separation. The first constraint equation we developed was:

$$travel_{ijk}(departure_{ik} + travelTime_{ij}) \leq arrive_{jk} \quad j = m+1, m+n, \forall i \in N, \forall k \in M, \tag{4.22}$$

However, this equation is nonlinear because two decision variables are multiplied together. In order to linearize equation 4.19, we utilized the method Ropke, Cordeau, and Laporte developed in [51]. We introduced a constant $a_{ij}$, which forces the algorithm to abide by the time constraints without multiplying decision variables together. Constant $a_{ij}$ is set before running the linear formulation by:

$$a_{ij} = \max\{0, late_i + observation_i + travelTime_{ij} - early_j\} \tag{4.23}$$

### 4.2.3 Solving Mathematical Program Optimal Solution

After we formulated the problem as a mixed integer program, we solved the problem using optimization software. Even though we knew that we could not solve the size of problems we were planning to solve, we wanted to test our formulation and test how large of a problem we could solve. A myriad of optimization software exists to solve Mixed Integer Programming (MIP) problems. Many noncommercial stand-alone programs or codes exist that can solve MIP problems. However, these generally lack reliability, technical support, fine-tuning of optimization parameters and suffer performance issues. Therefore, we chose to use commercial optimization software. The software that we chose as the optimization platform is ILOG [36].

ILOG is an optimization suite that supports linear programming, integer programming, mixed integer programming, mixed quadratic programming, constraint programming, scheduling, and modeling languages. There are several ways of running a problem in ILOG. ILOG provides a language that permits the matrix entries in the LP to be specified one at a time (which is very tedious) since it needs a very explicit formulation and write up of the whole problem. This is an extremely time-consuming, outmoded way of loading the problem and was not considered. Secondly, the dynamic libraries provided to interface the optimization solver to

any of the popular programming languages such as *C + +*, *C#*, or Java could be used. With this method, a code in either *C + +*, *C#*, or Java needs to be developed. Although this is a very powerful way of approaching the problem, it was not preferred due to the existence of an even more powerful alternative.

Apart from the two aforementioned methods, ILOG provides a special high-level optimization environment called ILOG OPL Studio, which makes it much easier and faster to model an optimization problem. Given the ease it brings, the ILOG OPL Studio was chosen as the implementation platform for the optimization problem. We will discuss the results of the optimization solver in Chapter 5.

## 4.3 Heuristic Methods Background

Many combinatorial optimization problems are computationally hard. Exact algorithms that deterministically find optimal solutions usually take too much time to be practical for large problems. In practice, these problems are solved approximately, using heuristics. Because of this, and reasons discussed in section 3.3.2.1.2, to solve problems of realistic size we need to explore heuristic algorithms for the USVOPP. Even though heuristics can solve the problem quickly, they do not guarantee optimal results or bounds on the distance from the optimal solution.

Many heuristics have been developed and tested to solve the OP and the VRPTW. In this section, we explore a couple different heuristic methods that exist that we could potentially implement to solve the USVOPP.

### 4.3.1 Approximate Dynamic Programming

Dynamic programming offers a cohesive approach to solving problems of stochastic control, which could solve the USVOPP. Essential to the technique is the optimal value function, which can be obtained via solving Bellman's equation [12]. For every possible m-tuple of USV initial positions, we determine the optimal path plan when the plan horizon is one time unit, and we store these optimal plans. Then for each m-tuple of USV initial positions, we determine the optimal path plan when the horizon is two time units. Each USV path in this plan

is either a move which requires two time units followed by an observation or a move which requires one time unit, takes an observation, and then performs a previously stored optimal plan for horizon=1. This process repeats until the plan horizon solved equals the plan horizon for the scenario. In this setting, a "state" gives the location (at some particular time) of all the USVs. Therefore, the number of points in the state space is the number of nodes in the static graph raised to the power "number of USVs"; we can see the curse of dimensionality explicitly.

Approximate dynamic programming was developed to alleviate the curse of dimensionality by considering approximations to the optimal value function, known as scoring functions that can be computed and stored efficiently. This approach can be applied to approximate a linear programming problem. The method generalizes the linear programming approach to exact dynamic programming [22].

However, the application of approximate dynamic programming appears to be technically difficult, so it is not pursued further in this thesis, but it is a possible area of future work.

### 4.3.2 Local Search Overview

Local Search algorithms, like approximate dynamic programming, came about as an attempt to circumvent the tractability issues confronted by the exact methods. They usually provide no bound on the optimality of their solutions, but generally provide a good solution quickly. The general concept of a Local Search method is to find and evaluate a solution within the feasible region, and then to evaluate neighbors to this solution. If a neighbor proves to yield a better objective function value, the algorithm moves to it and explores its neighbors. If not, the algorithm has found a local optimum, or a feasible solution that is at least as good as its neighbors. A typical Local Search algorithm will repeat this process several times with different initial starting solutions. What constitutes a "neighbor" to a solution can be interpreted differently, and the definition of a neighbor is often picked somewhat arbitrarily. Popular Local Search algorithms include Ant Colony, Genetic Algorithms, Neighborhood Search, Tabu Search, and Simulated Annealing.

#### 4.3.2.1 Ant Colony System

Bullheimer et al. [14] describe the Ant Colony System as a, "Distributed meta-heuristic for solving hard combinatorial optimization problems, first used to solve the TSP." The method was introduced by Colorni, Dorigo and Maniezzo, and is based on observed behavior of real ant colonies searching for food [19]. Specifically, ants communicate the information about food sources by marking the routes that lead to food with pheromones. Fellow ants can follow the pheromone trail and while following it, they will additionally mark it with new pheromones, thus attracting more ants. In result, routes that quickly lead to rich food sources will be reinforced. In solving a problem, simulated ants are searching the solution space, the quality and size of the food source correspond to the objective values that are being optimized, and adaptive memory plays the role of the pheromone trail. Ant systems are commonly used in production and route scheduling problems. An advantage to Ant systems is that they are easily updated for dynamic changes in the graph.

The application of an Ant Colony System appears to be technically difficult, so it is not pursued further in this thesis, but it is a possible area of future work to solve the USVOPP.

#### 4.3.2.2 Genetic Algorithms

Genetic algorithms (GA) are a class of adaptive heuristics based on the Darwinian concept of evolution of survival of the fittest. They were first developed by John Holland at the University of Michigan in 1975 [35]. Two main aspects of a GA that are problem specific are the genetic representation of the solution domain and the fitness function to evaluate the solution domain [66]. The most common way to encode a solution is as an array of binary numbers that represents them as a bit string. For example, if the range of values a variable can take is from zero to 32, we use a 5-bit string to encode the variable. Another popular encoding method is permutation encoding, which is more suitable for ordering problems. An evaluation function estimates the performance of a particular solution.

Genetic algorithms typically have the following structure Pseudo-Code:

1. Initialize a timer
2. Generate a random population
3. Evaluate fitness function
4. While not meeting the termination requirements do:
    5. Increment timer
    6. Select the fittest parents
    7. Recombine genes of selected parents
    8. Introduce mutations
    9. Evaluate fitness function
    10. Select survivors that will become the next generation

A GA initializes with a set of chromosomes referred to as initial population. Each chromosome represents a solution to the problem. The initial population is either randomly generated, which causes it to take longer for the algorithm to converge to the solution, or is generated using some form of heuristic, which takes less time to converge, but typically covers a smaller solution space and thus will converge to local optimal solutions.

A selection mechanism is used to select the potential parents based on their fitness computed by the evaluation function. Their offspring represent the subsequent generation. The selected parent chromosomes are then recombined via the crossover operator to create a potential new population. The next step is to mutate a few of the newly obtained chromosomes, in order to introduce a level of randomness that will prevent the GA from prematurely converging to a local optimum. A mutation is typically a random swap in the gene sequence, or a random denial of a bit if the chromosome was bit encoded. Then the new population is selected based on the fitness of the candidate chromosomes.

The genetic algorithm will then iterate through this process until a predefined number of generations have been met, or until a predefined level of fitness has been reached, or until there was no improvement in the population, which would mean that the GA has found an optimal solution. An application of GA to the USVOPP is beyond the scope of this thesis, but could be interesting future work.

#### 4.3.2.3 Neighborhood Search

Neighborhood search is a class of heuristics that seek to improve a feasible solution incrementally. It is applicable to many optimization problems. We refer partly to Ahuja, Dergun, and Orlin who give an overview of neighborhood search in [4].

Neighborhood search begins with an initial feasible solution, which may be obtained by randomizing or by another fast heuristic. Then, the algorithm searches the "neighborhood," a set of feasible solutions obtained by perturbing the initial solution. The solutions in the neighborhood include both better and worse solutions. As a rule of thumb, the larger the neighborhood, the more likely it contains better solutions, but at the expense of a longer search time. The algorithm then searches the neighborhood for a better solution, selects the best solution, and restarts using the best solution found as a current solution for generating another neighborhood. When there are no better solutions in the neighborhood, the current solution is called locally optimal, and the algorithm terminates. We refer to [2] for an extensive survey of theoretical and practical aspects of neighborhood search algorithms for combinatorial optimization problems.

We refer to a k-exchange neighborhood, which means that for k = 1, one target on one route is exchanged with one target on another route. For k = 1 or 2, the k-exchange neighborhoods can often be efficiently searched, but on average the resulting local optima may be poor. For larger values of k, the k-exchange neighborhoods yield better local optima, but the time spent to search the neighborhood might be too prolonged. The technique commonly known as the Variable-depth search method is a technique that searches several different values of k-exchange neighborhoods partially. The goal in the partial search is to find solutions that are close in objective function value to the global optima, while dramatically reducing the time to search the entire neighborhood. Typically, they do not guarantee to be local optima. In large-scale neighborhood search algorithms, we are interested in several types of algorithms for searching a portion of the k-exchange neighborhood. The Lin-Kernighan algorithm in [42] for the symmetric traveling salesman problem is a successful application of the variable-depth search.

#### 4.3.2.4 Tabu Search

Tabu search is a technique developed by Glover in 1986 that intelligently searches the solution space of a complex problem [28], [29]. The method works on the principle that the use of memory can help a search avoid being trapped in local optima and can help it look more closely at optima that may be sensitive to small changes. Tabu search explores the solution space in small steps and typically halts arbitrarily after some number of iterations, typically 1000 to 5000 iterations. The tabu search process is defined by an operation called a move. A move is an operation that changes any current solution. That changed solution is called the neighbor of the current solution since the changed solution is only one move away from the current solution. A move can be as simple as flipping a binary variable on or off or swapping two items in a sequence. During each iteration, the tabu search evaluates all, or some portion of, the neighboring solutions. All of the neighboring solutions comprise the neighborhood of the current solution. The tabu search picks the neighbor with the best objective function value, and that neighbor becomes the current solution for the next iteration.

As moves are executed, their particular attributes are registered on a tabu list, remaining there for a specified number of iterations, (the tabu tenure). The tabu list size and the tabu tenure are closely related and vary based on problem specifics. While tabu, the move cannot be chosen unless the tabu restriction is overridden by some useful goal, called the aspiration criterion. The basic purpose of the tabu list is to avoid revisiting previous solutions and force the search into unexplored areas of the search space. Altering the size of the tabu list helps control search features.

One of the drawbacks of tabu search is that there are several parameters used to specify a particular tabu search. Common parameters include the tabu list length and neighborhood size. These parameters can dramatically change tabu search performance. A tabu search that tunes these parameters as it searches the solution space is called a reactive tabu search. Battiti points out a drawback to tuning these parameters prior to the search process: "Either you need a lot of prior experience and knowledge about the problem or you need to do a lot of trial and error to find the parameter settings," [11]. Even though tabu search looks like a promising method, we leave it as possible future work.

#### 4.3.2.5 Simulated Annealing

Another common applied heuristic to large-scale problems is Simulated Annealing. Simulated Annealing is a generic method that builds on the Local Search method. As mentioned, Local Search is only capable of identifying whether or not a solution is a local optimum; in general, no guarantees on global optimality can be established when using Local Search. Simulated Annealing modifies the Local Search algorithm by occasionally allowing for a move to a worse solution. While counterintuitive, such moves give the algorithm an opportunity to escape from local optima that are not the global optimum. To explain the behavior of the algorithm, we give an example of a maximization problem like the USVOPP.

The algorithm initializes with a feasible solution, x, and picks a neighboring solution, y, at random. If the value of solution y is greater than that of x ($v(y) \geq v(x)$), the algorithm moves to solution y. If $v(y) < v(x)$, it will move to solution y with probability $e^{\frac{v(x)-v(y)}{T}}$, where T is a constant known as the temperature. It has been shown that the algorithm arrives at the global optimum with a very high probability after being run for a very long time at a very low "temperature," [38]. The low temperature causes the algorithm to take a long time to escape local optima, and it is for this reason that the algorithm must be run for a long time [38]. Because this is not practical, a compromise is to vary temperature with time, beginning with a larger temperature that allows the algorithm to explore more of the feasible area, and subsequently decreasing the temperature as the algorithm progresses. When the temperature is varied in this manner, it is known that the probability of the current solution being the global optimum equals 1 at the limit where time approaches infinity. Simulated Annealing is a general method, and while it has been shown to work very well for certain problems, it often underperforms special purpose methods designed for those specific problems. The application of simulated annealing is left as interesting future work.

## 4.4 Heuristic Algorithm Formulation

In this section, we explain the heuristic algorithms we developed to find approximate solutions to the USVOPP. While our algorithms do not solve the USVOPP to the global optimal solution, it gives reasonable solutions in a reasonable solving time.

### 4.4.1 The Problem Solving Methodology

We propose a new three-phase approach for solving the USVOPP. The methodology springs from the 4-phase heuristic Ramesh and Brown developed for the OP [47]. In the first phase, several initial solutions are created using a construction heuristic with different combinations of parameter values. In the second phase, we use an improvement heuristic to improve upon those solutions to cut away any slack in the initial solutions. In the third phase, we insert as many targets as possible in the routes. We compare the performance of our heuristic with a greedy construction heuristic.

A construction heuristic constructs a route by iteratively adding the "best" target similar to a nearest-neighbor routing. Construction heuristics are popular for solving routing problems very quickly. The construction heuristic we use for comparison with our three-phase approach to the USVOPP is a modification of a heuristic developed by Laporte and Martello [40] for the OP. In comparison to our algorithm, using a construction algorithm alone considers a much smaller portion of the feasible solution space. Therefore, the construction heuristic tends to have shorter computational running times in comparison with our algorithm, but generates solutions that are typically inferior.

### 4.4.2 Greedy Construction Algorithm

**Greedy Construction Algorithm Description:**

The construction algorithm we developed for the USVOPP was modified from the H1 algorithm that Laporte and Martello developed for the OP from the nearest neighbor algorithm Rosenkrantz, Stearns, and Lewis developed for the TSP [52]. It gradually extends a route $P$ by adding, at each iteration, a target according to a greedy criterion. The algorithm iteratively constructs a route for each USV in sequence. The target added to the route is selected by computing $\rho_{kn}$, the ratio of target $n$'s value, $value_n$, sum of the observation time, $observation_n$, upper time window time, $late_n$, idle time (if required), $idleTime_m$, and the additional travel time $travelTime_n$) incurred by $n$'s addition as in equation 4.21. The ratio is computed in this way to ensure targets that have positive features (such as shortest travel times, earlier time windows of

opportunity, shorter waiting times, and shorter observation times) have priority over targets that have negative features (such as longer travel times, later time windows of opportunity, waiting, and longer observation times). The next target added is the target that has the greatest value per unit of effort. The calculation of this ratio has a major effect on the routes produced by the algorithm. We equally weight the four criteria in this presentation, but a user could easily change the weights depending on his or her preference. The variable $ctime_j$ in the algorithm represents the current time for the USV $j$. The general scheme can be outlined as follows where $k$ is the initial location of each USV:

**Greedy Construction algorithm formal statement:**

1. Initialize global target available list, every target is available;
2. For each USV $j$ loop
3. Initialize: $P := \{k\}$, set $ctime = 0$, set available targets for particular USV from global target list
4. While feasible targets exist compute

$$\rho_n = \frac{value_n}{(observation_n + idleTime_n + late_n + travelTime_n)} \quad \forall n \notin P, \text{ if not feasible, } \rho_n = 0. \qquad (4.23)$$

A target is not feasible if visiting the target would violate a time window constraint, or the planning horizon.

1. Set the $\max\{\rho_n\}$ to bestValue, and the corresponding target to bestTarget, if bestValue $\geq 0$ then add bestTarget onto $P$
2. update *ctime*
3. remove targets with ratio $\rho_n = 0$ from the available target list for that particular USV
4. remove bestTarget from global target availability list

The procedure is illustrated in Figure 4.4. We assume this procedure represents an initial approach to solve the USVOPP.

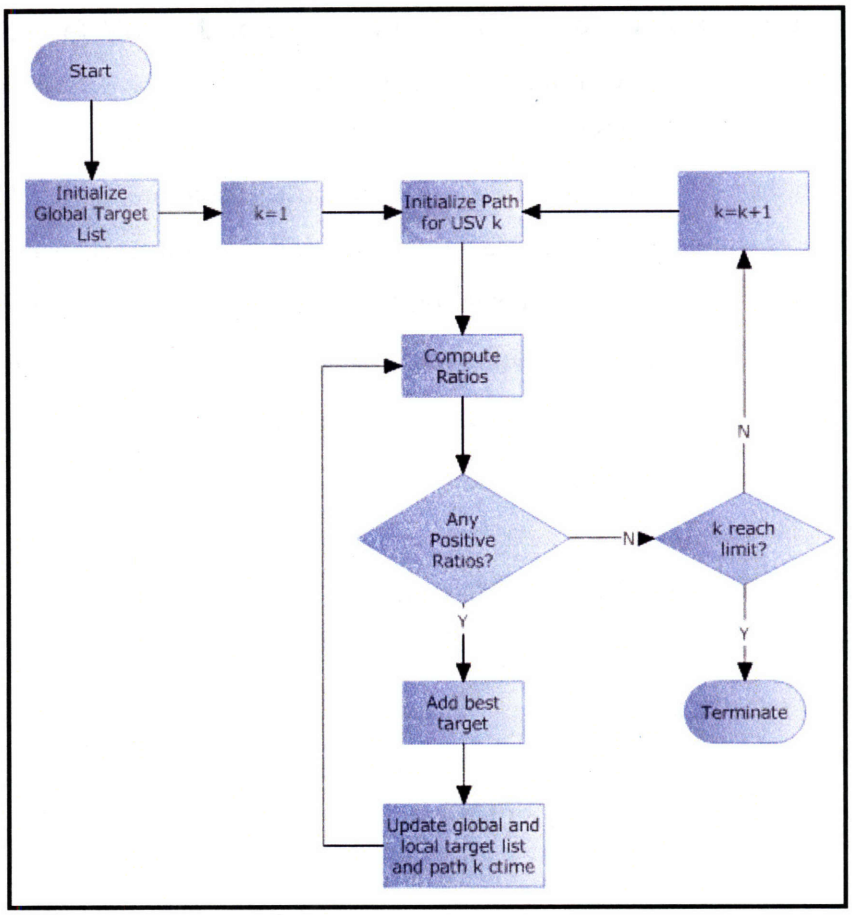



**Figure 4.4: Greedy Construction Algorithm Procedure**

### 4.4.3 USVOPP Local Search Heuristic

The formulation we present incorporates three phases: the construction phase, improvement phase, and the insertion phase.

#### 4.4.3.1 Construction Phase

USVOPP Construction Heuristic Description:

During the construction phase, we construct initial routes using a method like the greedy construction algorithm previously described. However, the difference being we analyze a target's value to each USV before adding the target to a route. The algorithm approaches the problem with a team mindset. After a target's value to cost ratio to each USV is determined, the

target with the highest value to cost is added to the route of the USV for which this ratio is biggest. For example, instead of routing the first USV, and then the second, we evaluate the ratios for each target for each USV all at once, and then add the target with the highest ratio across all USVs to the USV that values that target the most. Therefore, no USV routes steal any targets that are more valuable to other USVs. In addition, we construct many different initial solutions by changing the weight parameters on the observation time, $w1$, upper time window time, $w2$ , idle time , $w3$ , and travel time, $w4$ . This is done to cover more of the solution space and allow for the parameter settings that could potentially solve different types of datasets better than other parameter settings.

**Construction phase algorithm formal statement:**

1. For each set of parameter combinations *pc*
2. Set parameters w1, w2, w3, w4
3. Initialize Global Target List, every target is feasible
4. Initialize: $P_k := \{k\}$, $ctime_k = 0$ for all USVs
5. while feasible targets exist compute
    1. For each USV $k$ do

$$\rho_{kn} = \frac{value_n}{(w1 * observation_n + w2 * idleTime_{kn} + w3 * late_n + w4 * travelTime_{kn})} \quad \forall n \notin P_k,$$

if not feasible, $\rho_{kn}$=0. (4.24)

    2. select $\max\{\rho_{kn}\}$ if $\max\{\rho_{kn}\} \geq 0$ then add target $n$ onto $P_k$
    3. update $ctime_k$
    4. remove targets if $\rho_{kn} = 0 \ \forall k \in M$ and target $n$ from feasible target list

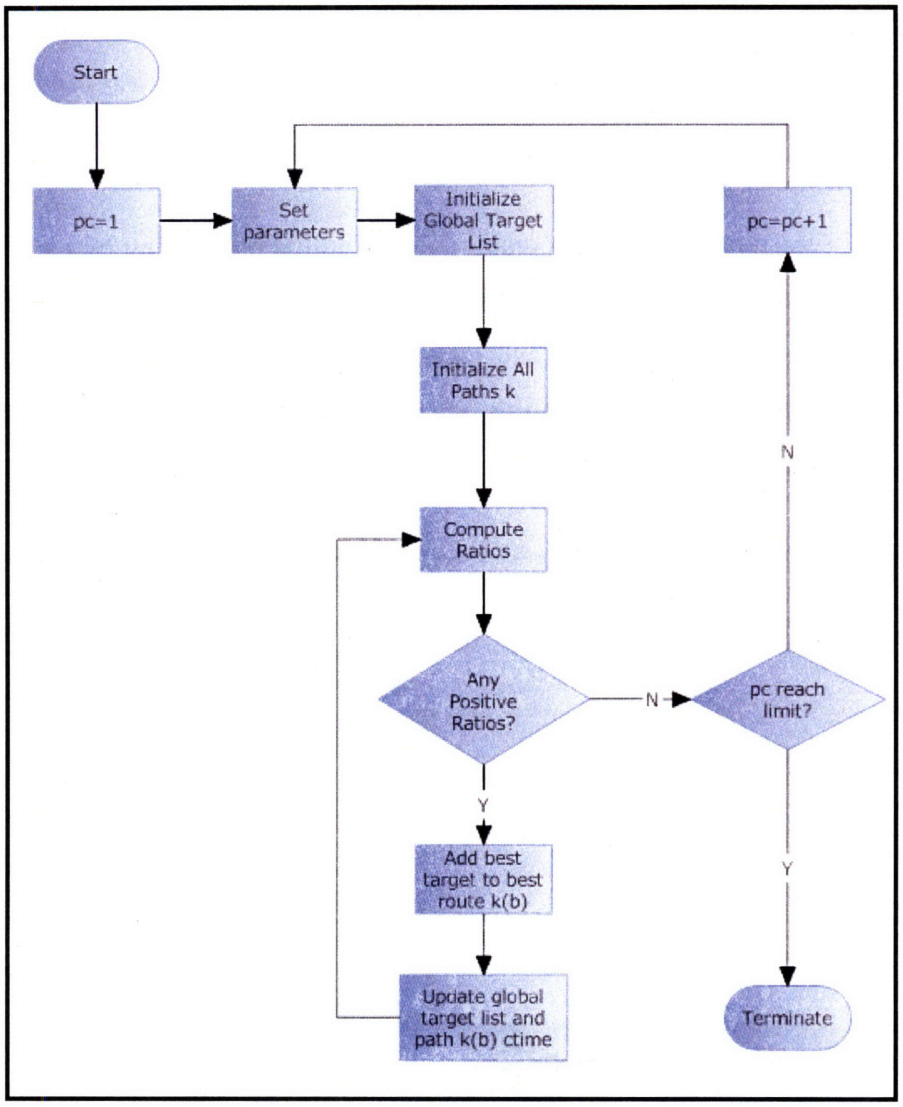



**Figure 4.5: USVOPP Construction Heuristic Procedure**

### 4.4.3.2 Route Improvement Phase

There are many different ways to modify a solution in order to obtain a new improved solution. Usually, the improvement step is embedded within a generic iterative improvement procedure of the following type.

Step 1. Modify the current solution to get a new improved feasible solution.

Step 2. Repeat Step 1 until no more improvement is possible (i.e. a local optimum has been reached).

We implement our improvement procedures after the initial routes are constructed. We utilize 3 route improvement methods in succession, an intra-route 2-Opt target swap, an inter-route target deletion and insertion, and an inter-route target exchange.

Intra-route Improvement Method:

2-Opt - is a local search procedure that creates a new solution by deleting two arcs and replacing them with two arcs in a route if it can reduce the time of the route and is still feasible, as illustrated in Figure 4.6. The indices $i$ and $j$ are in the order of the targets being visited in an original route. Figure 4.6 illustrates deleting arcs $(i,i+1)$ and $(j,j+1)$ and replacing them with arcs $(i,j)$ and $(i+1,j+1)$ or equivalently swapping target $i+1$ with $j$.

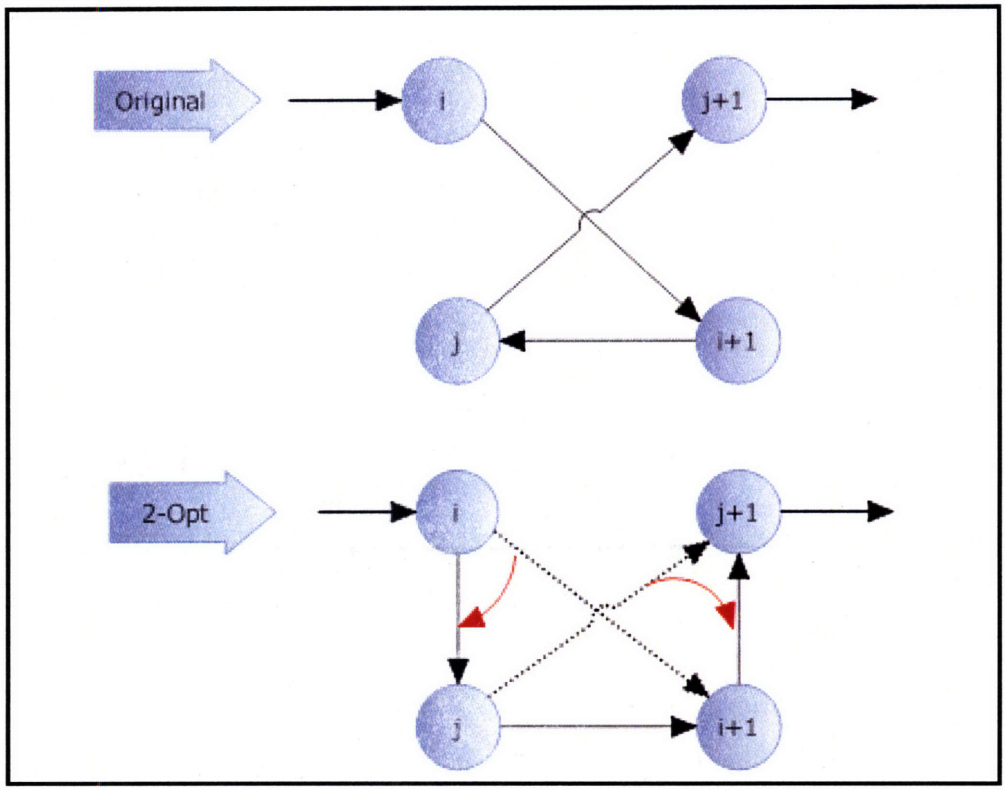



**Figure 4.6: 2-Opt Swapping Illustration**

However, Potvin and Rousseau [48] showed that classical 2-Opt exchange heuristics are not well suited for problems with time windows, because most swaps do not preserve the direction of the routes. Since targets are typically sequenced according to their time window's upper bound (those with the lowest upper bounds being the first to be serviced), reversing some portion of a route is likely to produce an infeasible solution. Knowing this, we limit the number of available

swaps for a route and thus decrease the run time of the algorithm. Only targets within $\lambda$% of the total number of targets on the route are eligible to be swapped. For example, if $\lambda$% =25%, and if a route has 10 targets, and we select the 4th target on the route, only targets 2, 3, 5, and 6 are eligible to be swapped with it; if we swap target 9 or 10 with 4, the new route is less likely to be feasible.

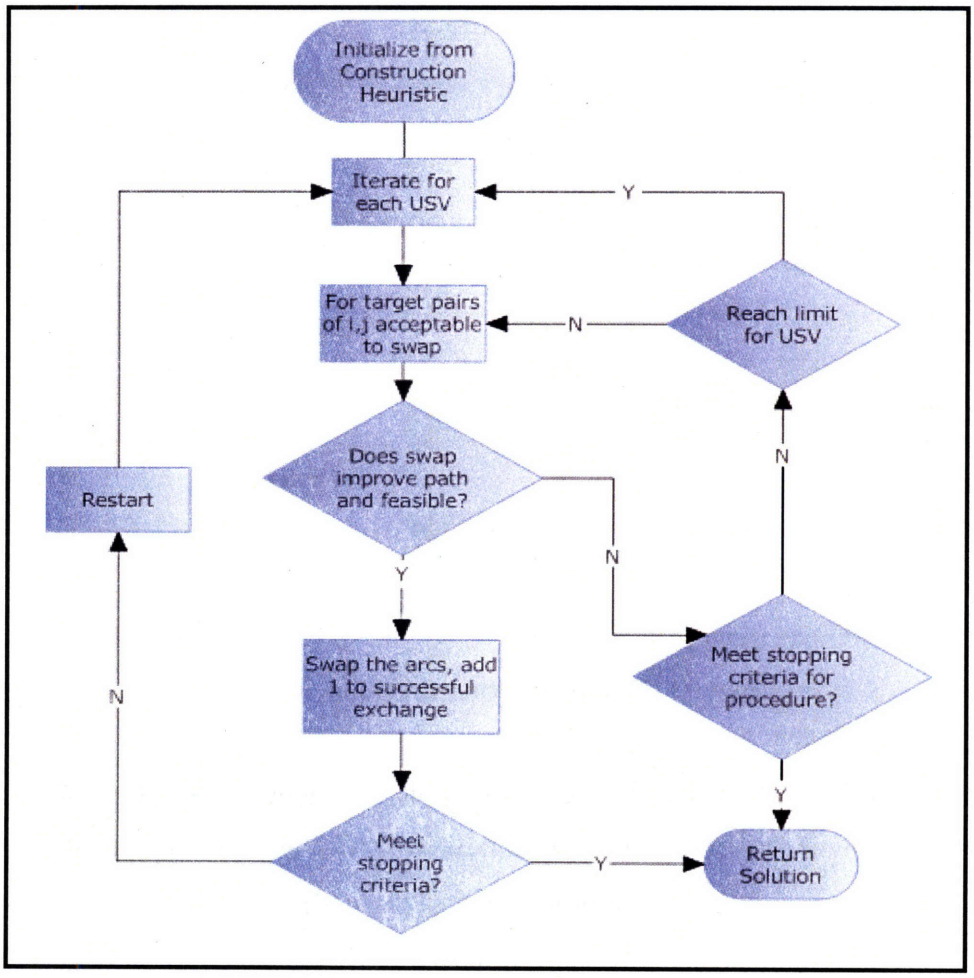



**Figure 4.7: Algorithm for 2-opt**

Inter-Route Improvement Methods:

Deletion-Insertion – is a method that deletes a target from one route and inserts the deleted target into another route as illustrated in Figure 4.8. Figure 4.8 illustrates deleting target *i* from the first route and inserting target *i* between *j*-1 and *j*. This is equivalent to deleting arcs (*i*-1,*i*), (*i*,*i*+1), and (*j*-1,*j*) and replacing them with arcs (*i*-1, *i*+1), (*j*-1,*i*), and (*i*,*j*).

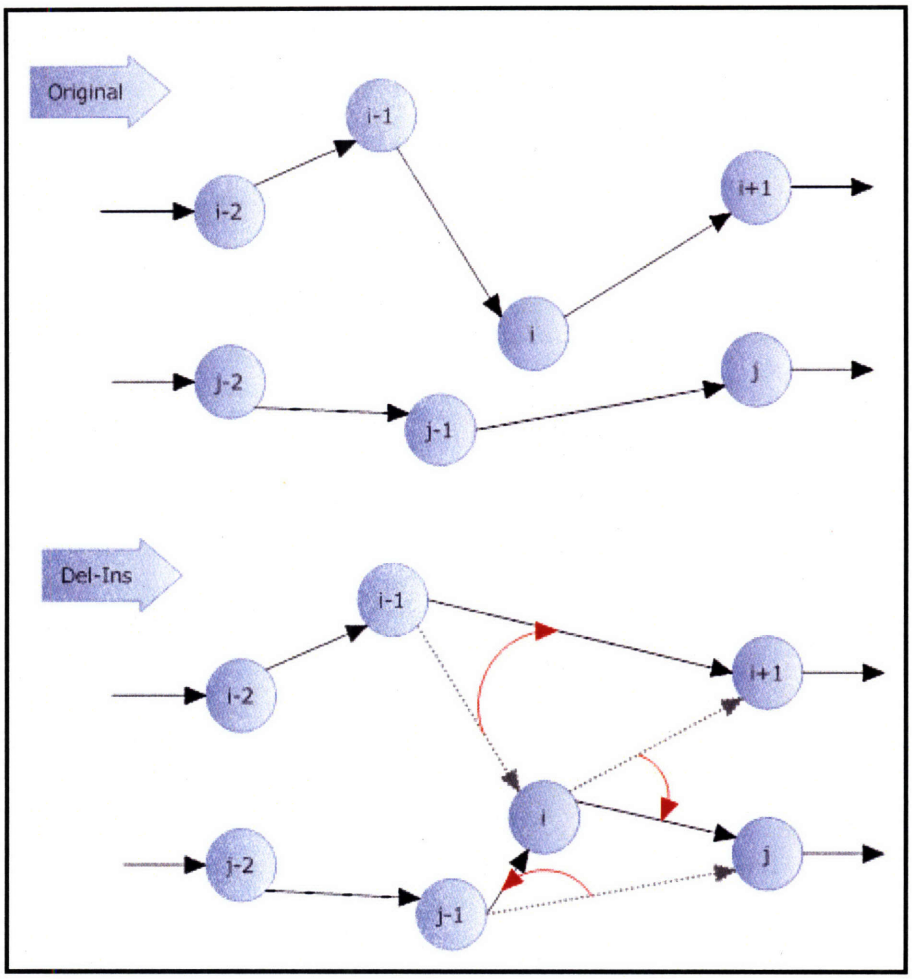



**Figure 4.8: Deletion-Insertion Procedure Illustration**

Similar to the way we modified the 2-opt, we limit the range of searching for an insertion location in a route by $\lambda$% in proportion for each route. Most likely, swapping outside of the $\lambda$% difference results in an unfeasible insertion, and not considering those swaps greatly reduces the run time for the algorithm. For example, if $\lambda$% =25%, if one route has 10 targets and another route has 20 targets, and we select the $4^{th}$ target on the first route for deletion, proportionally that equals the $8^{th}$ target on the second route. Target 4 from the first route is considered for insertion only after targets 3, 4, …, and 12 of the second route.

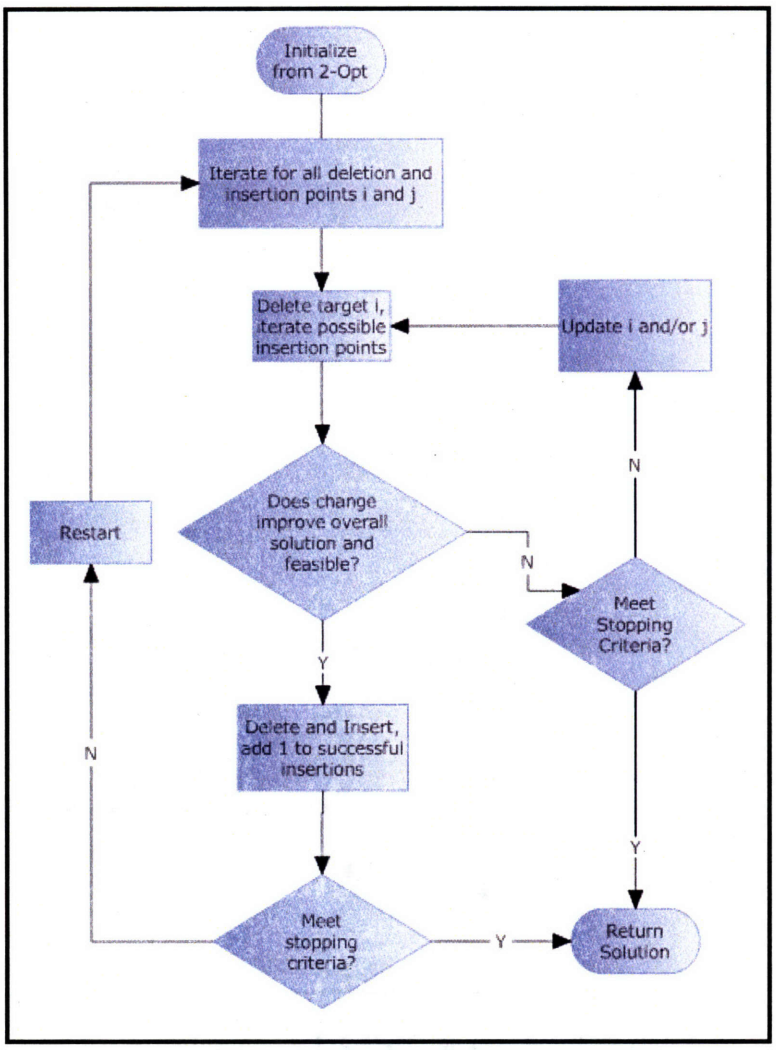



**Figure 4.9: Deletion-Insertion Procedure**

2-Exchange – is a method that deletes 2 arcs from two routes and replaces them with two new arcs on each route as illustrated in Figure 4.10. Figure 4.10 illustrates swapping two targets from different routes by deleting arcs $(i\text{-}1,i)$, $(i, i+1)$, $(j\text{-}1,j)$, and $(j,j+1)$ and replacing them with arcs $(i\text{-}1,j)$, $(j, i+1)$, $(j\text{-}1,i)$, and $(i,j+1)$. Equivalently, we can think of the 2-Exchange move as switching targets $i$ and $j$ between the two routes.

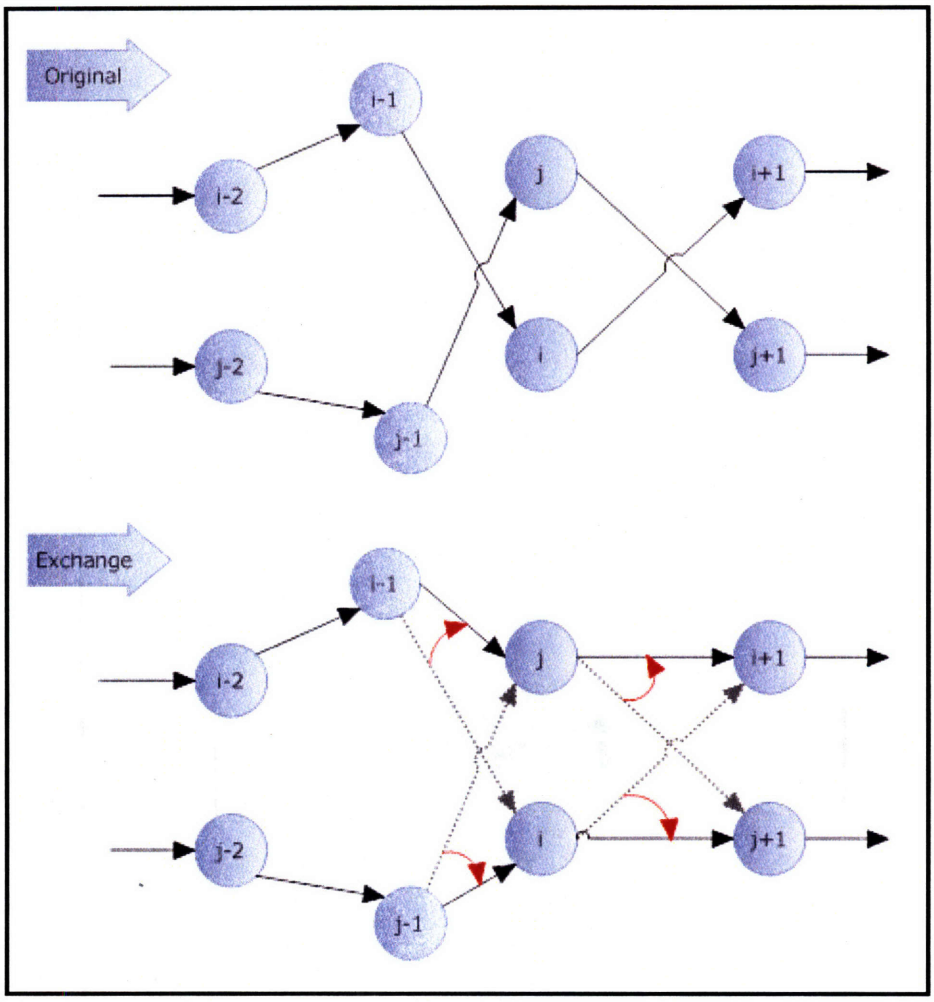



**Figure 4.10: 2-Exchange Method Procedure Illustration**

Similar to the way we modified the insertion-deletion procedure, we limit the range of swapping between routes to be $\lambda\%$ in proportion for each route. For example, if $\lambda\%$ =25%, if one route has 10 targets and another route has 20 targets, and we select the $4^{th}$ target on the first route, proportionally that equals the $8^{th}$ target on the second route. Target 4 from the first route is considered for exchange only with targets 3, 4, …, and 12 of the second route.

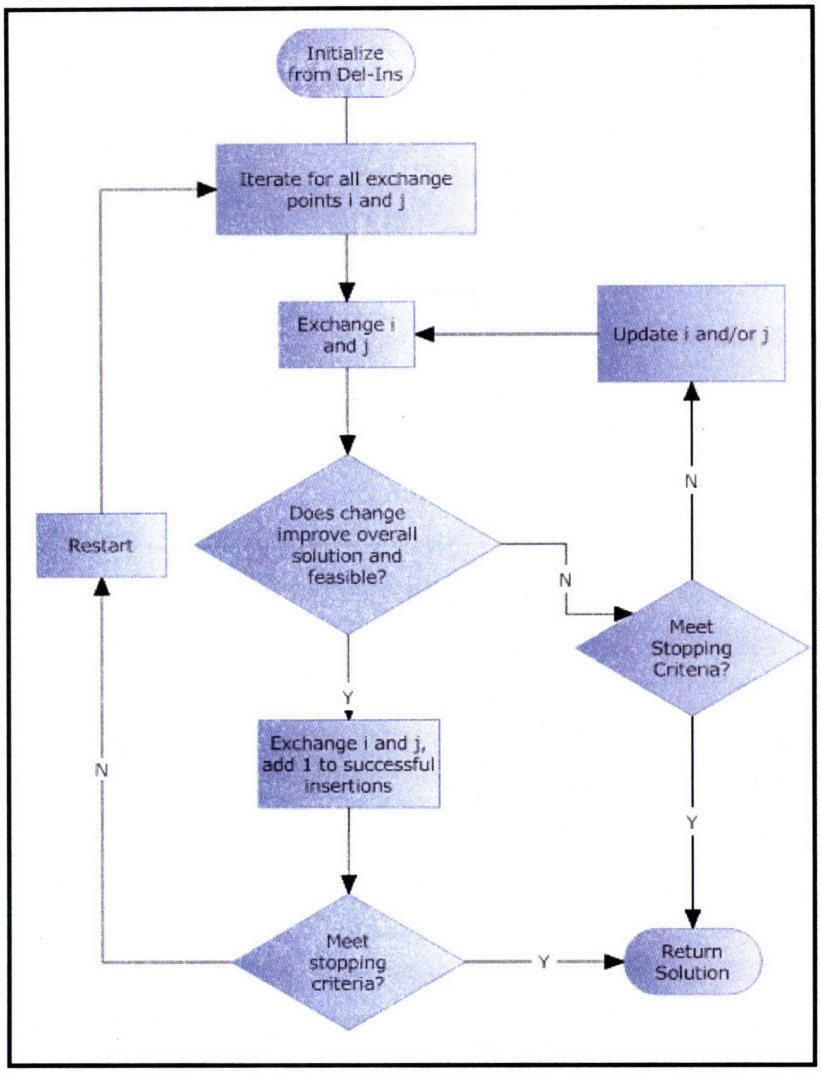



**Figure 4.11: Exchange Method Procedure**

The Improvement algorithm:

Based on the local search improvements, we have developed a new heuristic for the USVOPP that combines the three routines. The USVOPP Improvement heuristic:

Step 0: Initialize from construction heuristic

Step 1: For a set number of exchanges, or until no improvement exists, iterate through the intra-route 2-opt procedure.

Step 2: For a set number of exchanges, or until no improvement exists, iterate though the inter-route deletion-insertion procedure.

Step 3: For a set number of exchanges, or until no improvement exists, iterate through the inter-route exchange procedure, if the number of insertions reach a certain threshold, return to Step 1.

#### 4.4.3.3 Insertion Phase

After the improvement phase is completed, the insertion phase runs to insert as many unvisited targets as possible into the modified routes. In this phase, the unvisited targets are ordered according to their values. The target with the highest value in this set is attempted for a cheapest insertion by identifying the best location, by calculating the cost to value ratio, if more than one feasible location exists, to insert the target. If the target cannot be inserted without violating the constraints, then the target is discarded. This procedure is repeated for each remaining target in the order in which they have been arranged.

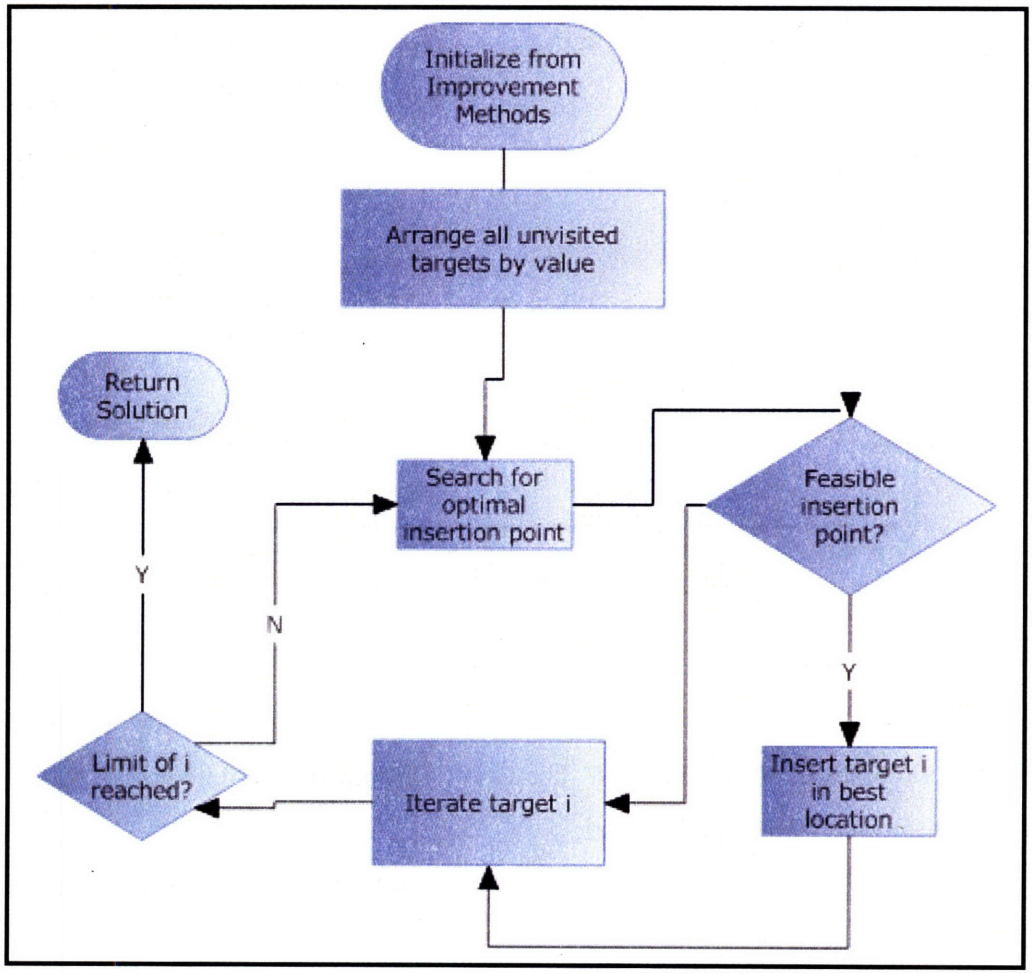



**Figure 4.12: Insertion Phase Procedure**

### 4.4.4 Total USVOPP Algorithm

We present the total solution for USVOPP in Figure 4.13.

Step 1: Construct multiple Initial USV routes as in Figure 4.5

Step 2: Improve initial solutions

Step 2a: 2-Opt procedure as in Figure 4.7

Step 2b: Deletion-Insertion procedure as in Figure 4.9

Step 2c: 2-Exchange procedure as in Figure 4.11

Step 3: Insert as many targets as possible into each USV route as in Figure 4.12

Step 4: Choose the best solution out of the different solutions produced:

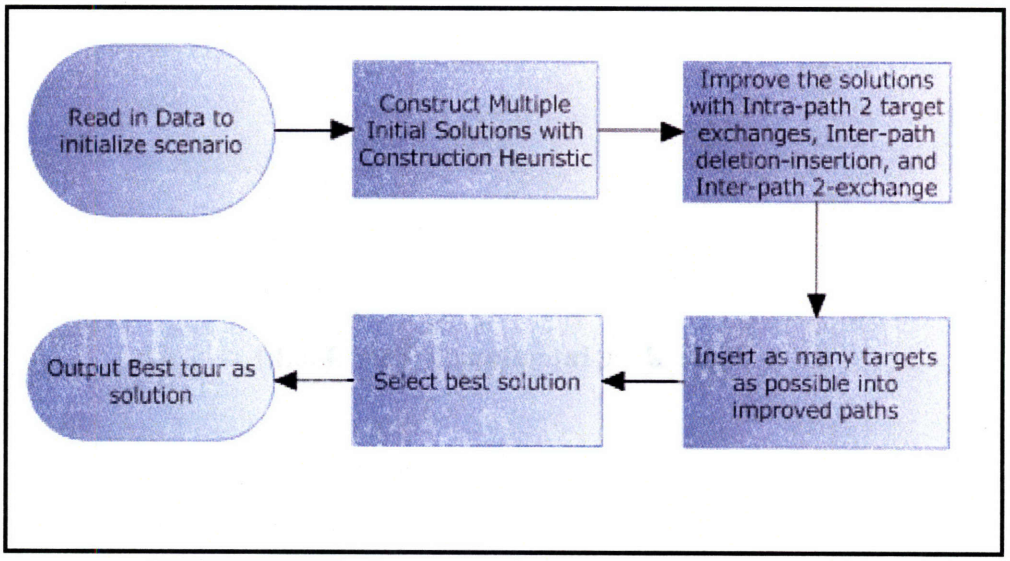



**Figure 4.13: Total USVOPP Algorithm**

We implement the algorithm in Java because Java is a readily available, powerful, and free programming language. In addition, the useful development software NetBeans IDE is free and makes the use of Java very user-friendly.

**[This Page Intentionally Left Blank]**

# Chapter 5

# Tests and Analysis

In our testing, we seek to establish the quality of our formulation and algorithms and evaluate the three-phase approximate algorithm (3PAA) for the USVOPP. In addition, we seek to fine-tune the algorithm to run efficiently for large datasets to allow for dynamic resolving. Unfortunately, we have not had the opportunity to test our algorithm using real data, value functions, and initial USV locations that arise from a real situation. Instead, we develop two test datasets that approximate the real life environments of our two scenarios. For each scenario, we compare the 3PAA with the simple greedy construction algorithm we developed. In addition, we compare the 3PAA with the exact algorithm that ILOG uses, and the simple greedy construction algorithm for smaller and randomly generated datasets. The topics of this chapter are presented in the following order:

1. Test Datasets and Environment
2. Metrics and Parameters
3. Hypotheses
4. Evaluation Tests and Results
5. Summary

## 5.1 Test Datasets and Environment

This section describes the software and hardware we used in the development of datasets and testing. The test conditions are summarized as follows:

1. We performed the tests on a Draper Laboratory personal computer with dual core (3.60 MHz, 3.59 MHz Intel Pentium 4) processors with 1.0 GB of memory. The computer was equipped with the Windows XP operating system. No other non-essential application programs were running when we conducted the tests.
2. We implemented the exact algorithm in ILOG OPL Studio v5.0 Integrated Development Environment (IDE) that uses ILOG CPLEX solver v.10.1.0 to solve the MIP.
3. We implemented the greedy and 3PAA algorithms in Java using the NetBeans IDE version 5.5.
4. The test datasets were generated in MATLAB.
5. All CPU times are given in seconds.
6. We computed an average over several runs of the same problem to remove some possible random noise in the test runs.

ILOG uses the optimization solver called CPLEX to find the solution to the USVOPP. CPLEX first uses preprocessing methods to reduce the size of the problem without eliminating the optimal solution. Then it uses the Simplex Method to solve for the linear relaxation of the problem. Finally, it finds the optimal integer solution via its branch-and-bound algorithm. In very large problems, the branch-and-bound tree becomes too large for the RAM of the machine and the program terminates, returning the current integer solution that is suboptimal.

The two scenarios and the pseudo-random datasets were generated with Matlab software. A scenario is specified by a dataset whose layout was discussed in Chapter 4 section 4.1. The

Matlab random number generator is used to generate datasets for all of the small problems. We used a uniform distribution to generate pseudo-random USV initial locations, target locations, target values, time windows, and observation times.

To the best of our knowledge, no generalized test database exists (such as the TSPLIB for TSP problems) for team orienteering problems with time windows. If there were such a database, we could use it to test our algorithm against known optimal solutions, known best solutions, or other heuristics.

## 5.2 Metrics and Parameters

In our testing of the formulation, we desire to compare solutions generated by the 3PAA with the solutions of the greedy algorithm and in some cases, the exact algorithm. In order to compare the algorithms objectively, we require a set of metrics with which to quantify the solution quality.

### 5.2.1 Metrics

We use two measures of performance to distinguish between solutions, the objective value of the solution and the CPU computation run time. The objective function value is important because it represents how much value is observed by following a generated solution, the ultimate goal in solving the USVOPP. It is easy to quantify and to compare across each method. Although the objective value is important, if the algorithm takes too long to generate the solution, it is irrelevant. Comparing the CPU computation time along with the objective value gives us a total picture for the algorithm performance.

When all the results are tabulated, we calculate both average values and variances to the metrics. While the average for the metrics is important, it does not provide a complete picture of the results of the datasets. By including the variance about these average values, we can develop a better sense of the likelihood of achieving these particular metrics on future trials with similar initial characteristics. We assume that a lower variance is preferable to a higher one, and that a solution with a lower mean value could still be preferable to one with a higher mean value if it has a lower variance about this mean.

### 5.2.2 Selecting values of the parameters

We adjust several parameters in the construction and improvement phase to produce different solutions. Changing the values of the parameters used in the 3PAA affect the run time performance and the solution quality. In the construction phase of the 3PAA, targets are added to the schedule in order of priority. The priority of a target is the weighted sum of 4 quantities. The weights given to these quantities are:

- $w1$ Observation Waiting Time Factor (Increasing this factor's weight increases the relative value of targets with lower observation times)
- $w2$ Late Time Window Factor (Increasing this factor's weight increases the relative value of targets approaching time window deadlines or their *late* values)
- $w3$ Travel Time Factor (Increasing this factor's weight increases the relative value of targets that are closer to the current USV locations)
- $w4$ Wait Time Factor (Increasing this factor's weight increases the relative value of targets that require less idle time)

These four weights are adjustable parameters. To find reasonable values for these weights, we performed a set of experiments, using 80 randomly generated datasets of different sizes. We tested a range of values for each weight between 0, and 1. The sum of the four weights always equals one. Setting a parameter weight to zero is the equivalent of turning off that factor. Setting all parameter weights to 0.25 is the equivalent of the greedy heuristic's weight factor settings that all factors are equal. Setting a parameter weight to 1 is the equivalent of using only that factor. We tested 32 different combinations of parameter settings for 80 randomly generated datasets of different sizes. We found a few sets that consistently constructed routes whose objective function value was the largest and a few sets that we found to never construct routes that led to the best final solution. Here is the list of the best eight settings we found:

| Setting Number | w1 | w2 | w3 | w4 | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | Late time window only priority |
| 2 | 0 | 0 | 1 | 0 | Travel time only priority |
| 3 | 0 | 0 | 0 | 1 | Wait time only priority |
| 4 | 0.25 | 0.25 | 0.25 | 0.25 | All factors equal priority |
| 5 | 0 | 0.5 | 0.5 | 0 | Late time window and Travel Time only priority |
| 6 | 0.4 | 0.1 | 0.1 | 0.4 | Observation and wait times higher priority |
| 7 | 0.4 | 0.1 | 0.4 | 0.1 | Observation and travel times higher priority |
| 8 | 0.3 | 0.1 | 0.3 | 0.3 | Observation, travel and wait times higher priority |

**Table 5.1: Best Parameter Settings**

In every trial of the 80 we ran, one of these eight parameter settings ended up as the best setting. Therefore, we limit the 3PAA to only considering these eight combinations to ensure the algorithm executes faster without losing good initial starting locations.

For the improvement phase, we use the parameter $\lambda$ to represent the percentage of the route that the algorithm searches for available exchanges. Figure 5.1 illustrates an example how increasing the parameter $\lambda$ increases the number of iterations for an inter-route exchange. The red circle with a 3 on Route 1 is the current starting target. For $\lambda = 0.1$, the black solid line between Route 1 target 3 and Route 2 target 8 represents the only possible exchange. For $\lambda = 0.25$, the dashed blue lines between Route 1 Target 3 and Route 2, targets 7, 8, and 9 represent the three possible exchanges.

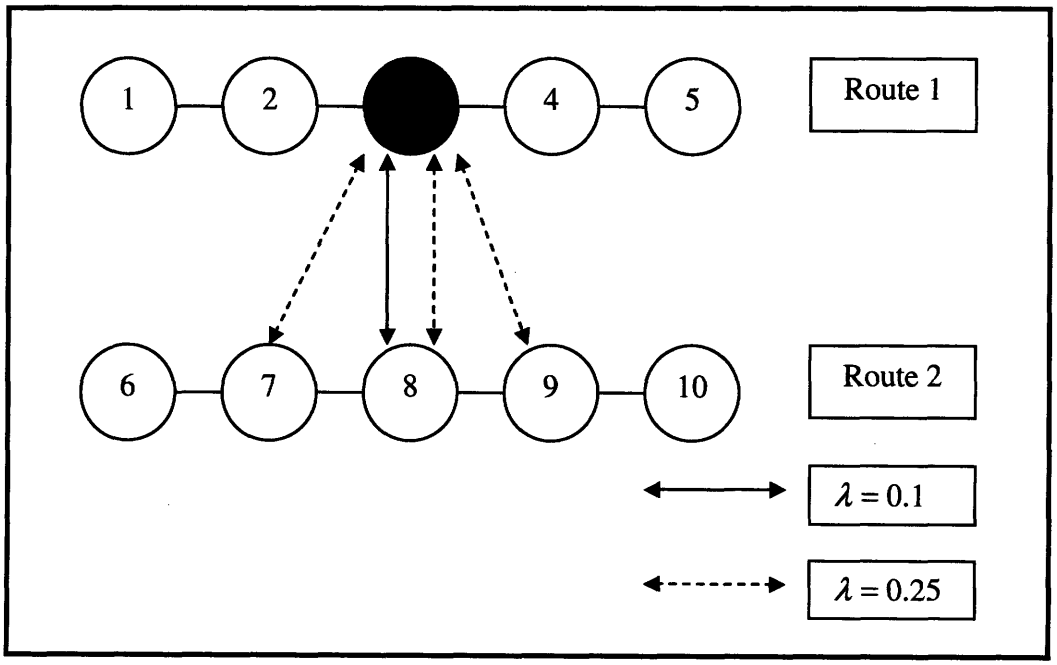



**Figure 5.1: Lambda Diagram**

Generally, increasing $\lambda$ increases the run time of the algorithm, but also improves the solution quality. However, because of time windows, excessively increasing $\lambda$ dramatically increases run time without increasing solution quality due to diminishing marginal returns for searching more options that are less likely to improve the solution. In general, making more time efficient routes increases the number of available insertions, but in some cases, a more time efficient route forces targets, which could have been inserted before improvements, infeasible to insert after improvements. Increasing lambda never increases the travel time of the USV routes, but because of time window constraints, occasionally changed which targets were could be inserted and decreased the objective value. We seek to find the "best" value for $\lambda$ that provides the highest objective values with reasonable run time.

We tested different values between 0.0 and 0.5 for $\lambda$ over different datasets and recorded the objective value and the run time performance.

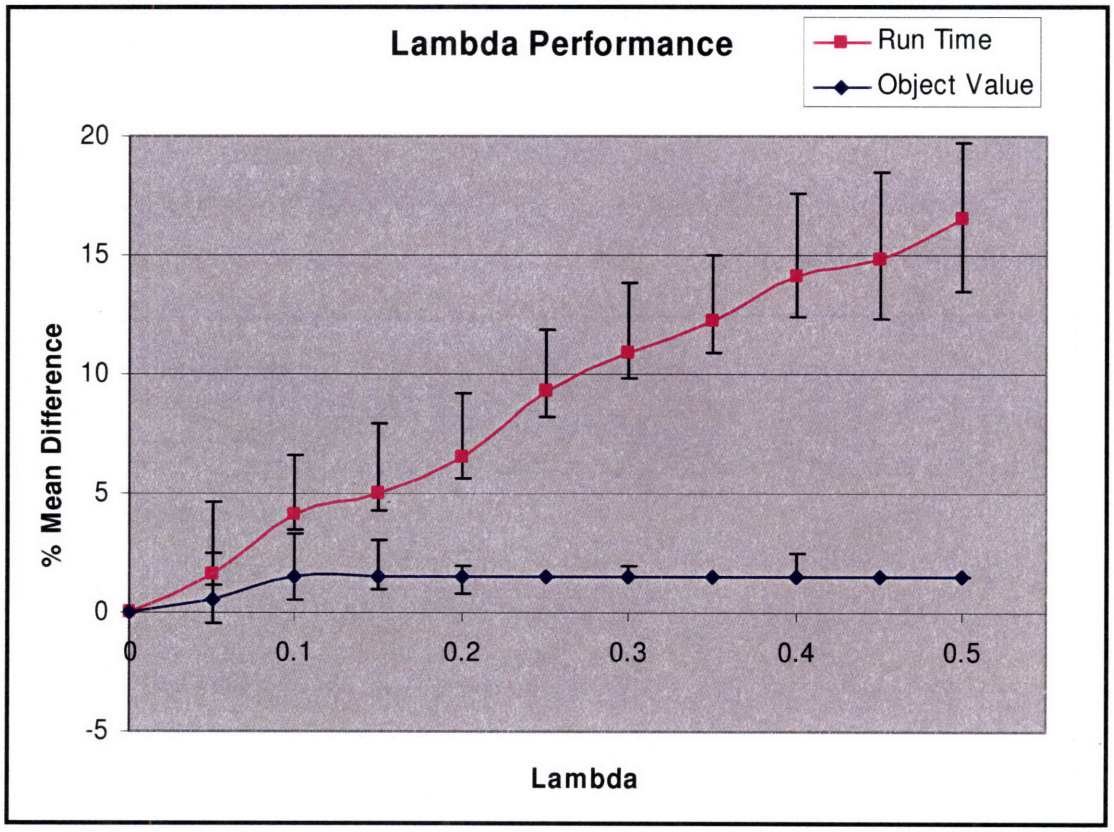



**Figure 5.2: Lambda affect on Objective Value and Run Time**

Figure 5.2 illustrates the diminishing marginal returns of increasing $\lambda$. Therefore, for the scenario data, we set lambda permanently at 0.1, because higher values of $\lambda$ increase the run time without increasing the objective value.

## 5.3 Hypotheses

Our interest in testing is to investigate the quality of the 3PAA. We expect that the 3PAA obtains better objective function value than a greedy construction algorithm, yet does not take significantly longer run time in large-scale problems. These expectations motivate the experiments, and form the hypotheses that we test in our experimentation listed here:

Hypotheses:

1. The average run time required for the exact algorithm, as formulated in section 4.2.2, and solved by CPLEX, increases exponentially with the number of targets.
2. On average, as an algorithm's objective function value increases, run time performance decreases
3. On average, substituting Spartan Scout USVs for OASIS USVs does not significantly improve objective function value in HAB Scenarios
4. On average, substituting Spartan Scout USVs for OASIS USVs does significantly improve objective function value in Hurricane Scenarios

## 5.4 Evaluation Tests

We now step through the experiment results. Each experiment is contained within its own section. The sections provide a motivation for the scenarios, present the hypotheses tested and scenarios used, present the output, and provide analysis for the experiment.

### 5.4.1 Exact Algorithm Performance

The first experiment is intended to test the exact formulation and its solutions and run-time performance. We tested our hypothesis that the exact algorithm's run time is an exponential function of the number of targets. We performed a series of randomly selected runs to obtain an average run time for each value of targets. Through our trials, the exact algorithm's average run time did increase exponentially. The exact algorithm's data is included in Table 5.2. In addition, the algorithm maxed out our computer's memory when we attempted to solve problems with 40 nodes and 2 USVs because the ILOG software returned an out-of-memory error. Problems with 35 nodes and 2 USVs took the solver about a day and a half to solve. The number of USVs did not have the same affect increasing the run time as did the number of targets. Increasing the number of USVs for a small number of nodes actually decreases the run time because it is more likely to reach every target with more USVs available. The longest run times occur when most of the targets are reachable, but only half the targets could feasibly fit in the final solution.

Figure 5.3 illustrates the exact algorithm logarithm of average run-time of the pseudo-random datasets as a function of the number of targets. If this function were concave up, then the exact algorithm run-time performance would be worse than exponential. However, we have a relatively straight line and cannot conclude that the performance is significantly better or worse than exponential. We found that the run-time performance increases exponentially. In addition, the variance of the run time also increases exponentially as the number of targets increase.

| Targets | Solve Time (secs) | Standard Deviation |
|---|---|---|
| 5 | 0.91 | 0.05 |
| 10 | 4.57 | 0.97 |
| 15 | 26.32 | 3.23 |
| 20 | 635.19 | 159.46 |
| 25 | 7254.48 | 2071.428571 |
| 30 | 86960.72 | 29986.2069 |

**Table 5.2: CPLEX Run-Time Data**

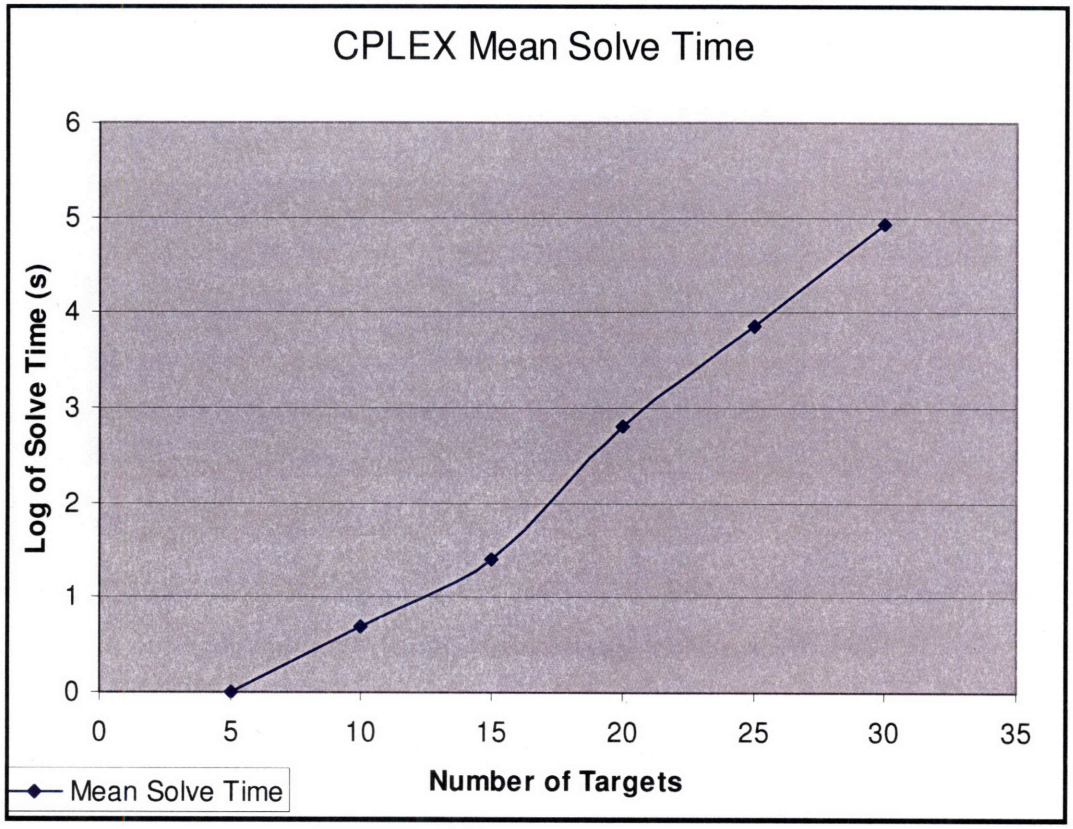



**Figure 5.3: Exact Algorithm Run Time Performance**

We found that generally, the run-time performance of the exact algorithm spends the bulk of the time proving that the solution is optimal. Figure 5.4 illustrates that the bulk of the time between finding an integer solution and the end time is spent proving that the last integer point is optimal. The red line is the best possible feasible relaxed solution (non-integer), and the green line represents the best current integer solution. When the algorithm finds a better integer node, a yellow square marks the point in time. Figure 5.4 illustrates that the best integer is found at approximately 58 seconds, but does not prove this to be the optimal solution until approximately 5 minutes later.

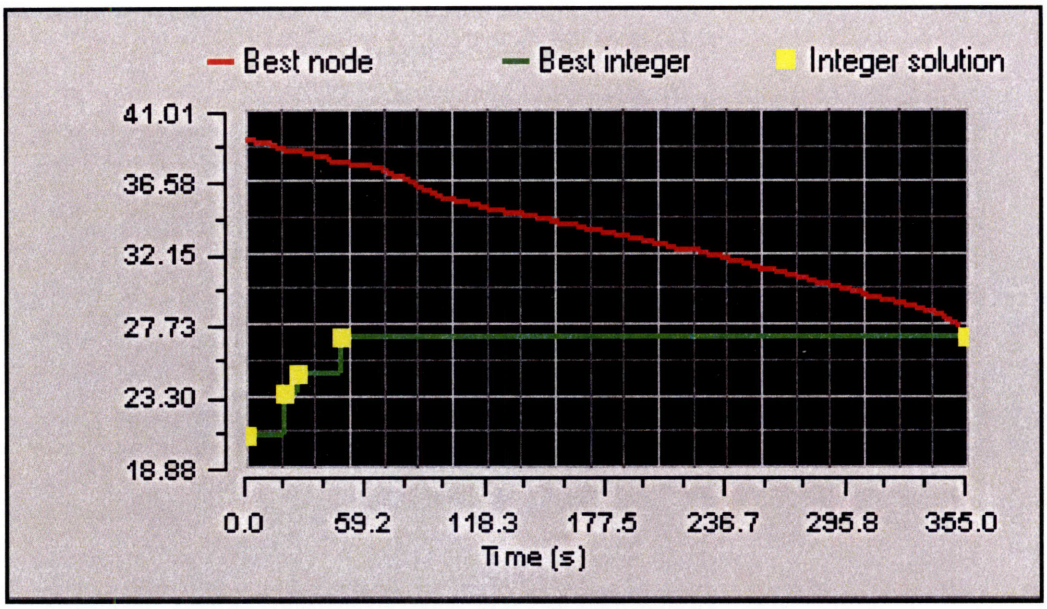



**Figure 5.4: Example Run-time of Exact Algorithm**

### 5.4.2 Comparing 3PAA to Optimal and Greedy Algorithms

We measured the solution quality of the heuristics as the percentage deviation from optimality, which we defined as the *shortage*. The shortage is calculated as follows in equation 5.1.

$$Shortage = \frac{optimal\_value - solution\_value}{optimal\_value} \tag{5.1}$$

#### 5.4.2.1 Solution Quality Test of 3PAA for Random Datasets

We evaluate the performance of the 3PAA using a group of small to medium-sized Matlab pseudo-randomly generated cases. For the random cases, we randomize the initial USV locations, initial target locations, target values, required observation times, and time windows. In the name used to refer to any generated case, the numerical data specifies the number of USVs and the total number of targets. For example, "b2t10" has 2 boats and 10 targets. The number of targets in the test sets varies from 10 to 25. In total we used 20 test cases, 5 random test cases for each of the 4 sizes of targets. In Table 5.3, the first column is a list of the test cases that we used, sorted by the number of targets. The second column specifies the optimal value for each case.

The third column is the value generated by the 3PAA, and the last column is the relative deviation from optimality, or the shortage. The relative performance of the 3PAA is good on these test cases.

| Problem | Optimal | 3PAA | Shortage |
|---|---|---|---|
| b2t10 | 9.2188 | 9.2188 | 0.00 |
| b2t10a | 13.7494 | 13.7494 | 0.00 |
| b2t10b | 11.8610 | 11.8610 | 0.00 |
| b2t10c | 21.1330 | 21.1330 | 0.00 |
| b2t10d | 10.0210 | 10.0210 | 0.00 |
| b2t15 | 29.2030 | 29.2030 | 0.00 |
| b2t15a | 27.8010 | 27.3556 | 0.02 |
| b2t15b | 21.0560 | 21.0560 | 0.00 |
| b2t15c | 25.6560 | 22.5743 | 0.12 |
| b2t15d | 14.6240 | 14.6240 | 0.00 |
| b2t20 | 36.0450 | 35.7890 | 0.01 |
| b2t20a | 26.8440 | 24.8236 | 0.08 |
| b2t20b | 19.4005 | 18.5042 | 0.05 |
| b2t20c | 32.1670 | 28.6075 | 0.11 |
| b2t20d | 32.2640 | 27.4768 | 0.15 |
| b2t25 | 27.6198 | 24.1322 | 0.13 |
| b2t25a | 24.2310 | 21.1053 | 0.13 |
| b2t25b | 34.6874 | 33.0090 | 0.05 |
| b2t25c | 32.1456 | 32.1456 | 0.00 |
| b2t25d | 47.8340 | 38.1902 | 0.20 |

**Table 5.3: Shortage Results of 3PAA**

The average shortage over all cases is 0.05, or 5% from optimal, with a variance of 0.0042 or approximately 0.5%. In addition, for the problems involving 10 and 15 targets, the 3PAA found the optimal solution in 80% of the cases. For each case, the standard 3PAA settings were used. The standard settings include iterating through the 8 initial parameter settings, setting $\lambda = 0.25$, setting no limitations on the number of switches the improvement phase can complete, and no limitations on the number of insertions. The value reported for the 3PAA is the value generated from the best final solution that the 3PAA generated from the 8 different parameter combinations.

### 5.4.2.2 Relative Performance of 3PAA

Next, we compare the results of the 3PAA with the greedy heuristic that constructs routes one USV at a time. Figure 5.5 provides a comparison of the 3PAA on the test cases with the greedy construction heuristic algorithm. On average, the greedy construction algorithm (GCA) had a shortage of .23 or 23% and a variance of .0233 or 2.3%. Therefore, on average, the 3PAA improved the solution quality by 77% on these test cases without increasing the run time by more than a couple hundred milliseconds, which is hardly noticeable. Each of the heuristics completed every test case in less than one second of run time.

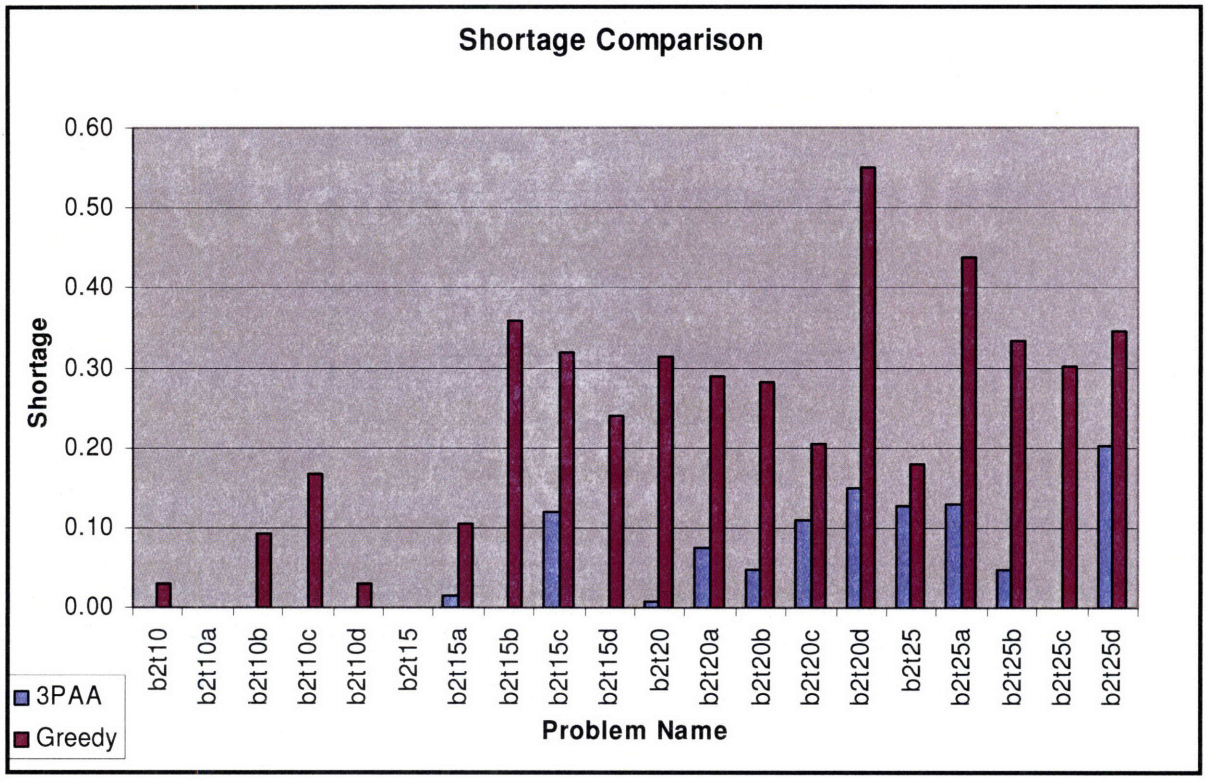



**Figure 5.5: Shortage Test Results for 3PAA and Greedy Algorithms**

As illustrated by Figure 5.5, the 3PAA performs significantly better than the greedy heuristic for relatively small pseudo-random datasets.

### 5.4.2.3 Run-time Performance of 3PAA

Next, we evaluated the run-time performance of the 3PAA using larger pseudo-randomly generated test data. For each test set with targets ranging from 10 to 500, we generated five random datasets. We ran each dataset 5 times to limit the random noise between trials. The random noise includes whatever background tasks the operating system is performing behind the scenes. We measure the run-time of each trial by subtracting the internal system clock time stamp within the Java code at the beginning of execution, from the time stamp at the end of execution, before the results are printed to the screen. We conducted the test to see how the run time of the 3PAA with standard settings varied as a function of the number of targets. Figure 5.6 illustrates the average computational time versus the number of targets.

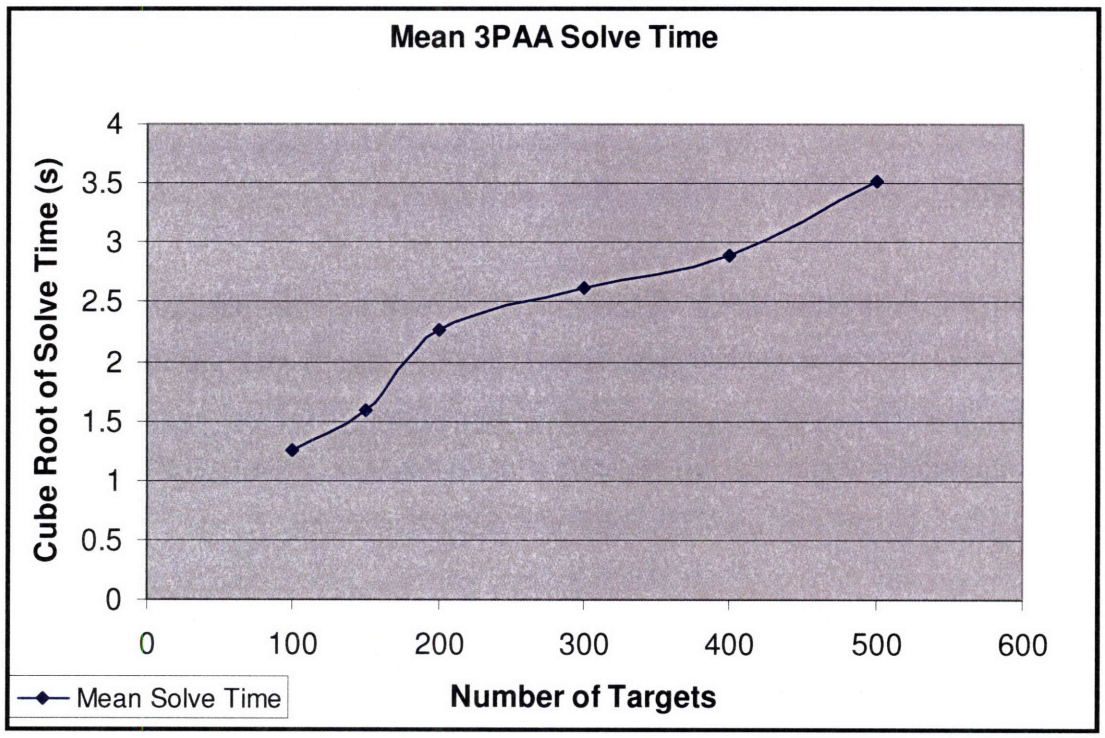



**Figure 5.6: Run-Time for Standard 3PAA**

Figure 5.6 illustrates how the average run-time performance of the 3PAA with standard settings has cubic growth. However, the exact algorithm grows in hours, the 3PAA grows in seconds. In the largest cases, the 3PAA with standard settings solved a 10,000-node problem

with 24 USVs in approximately 17 hours and 18 minutes. The GCH solved the same problem in 22 seconds. Computing the relative performance, the 3PAA improved the objective function value by 41%. In the most extreme case, the GCH solved a 16,275 node problem with 24 nodes in about 10 minutes. However, of the 10 minutes, 9 minutes and 30 seconds represented the time to read the data and construct the travel time matrix, and the algorithm executed in 30 seconds. Increasing the size of the problem beyond 16,275 nodes caused the computer to freeze because it utilized 100% of our computer's RAM.

### 5.4.3 HAB Scenario

Now, we explore the results of the 3PAA on the realistically sized HAB datasets. For the following examples, the number of targets ranged from 500 to 3000. We chose this range for the number of targets to explore because changes in the algorithm in this range dramatically change the run time. For smaller problems, changes in the algorithm only have minor effects on the results. The HAB Example presents the computational experiments and evaluation results.

**HAB Example 1:**

Initialization:

Figure 5.7 illustrates an HAB scenario where an HAB has been identified and 8 USVs in the area are called to monitor the situation. The white dots in this figure represent the initial USV locations and the HAB is located in the small colorful region in the lower left side of the graph. The colors represent the value of observations, with blue being no value, red being high value. The vertical axis is North-South, and the horizontal axis is East-West.

HABs typically move with the tidal currents, which we have assumed away, and expand very slowly. Figure 5.8 illustrates the projected 4-day path of our example HAB. The graph is focused only on the region near the HAB, as opposed to Figure 5.7 that illustrates, not only the HAB region, but the entire region, which includes the initial locations of the USVs. The color within the graph represents the value of the targets as previously described in section 4.1.2.2. The planning horizon is 96 hours. The value for the each target changes for each day, and is represented in the data by independent data points with the same location, but different time windows and target values.

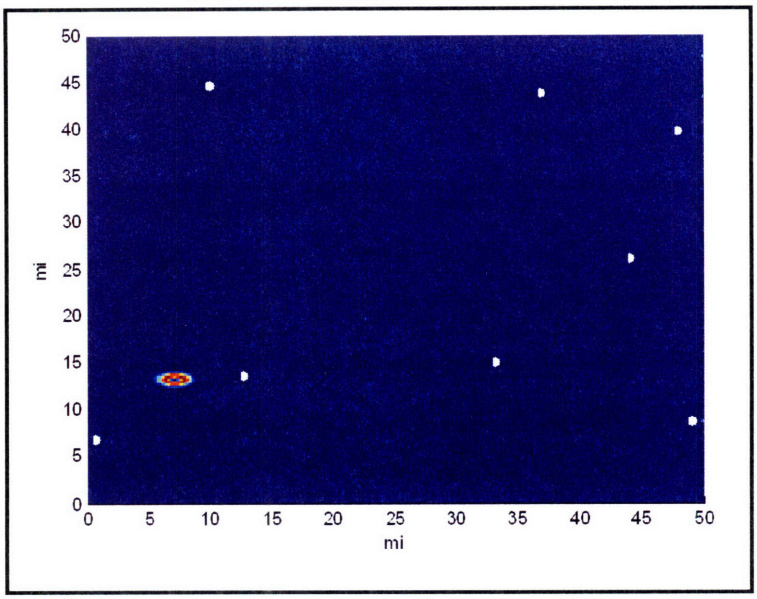



**Figure 5.7: Initial HAB Scenario**

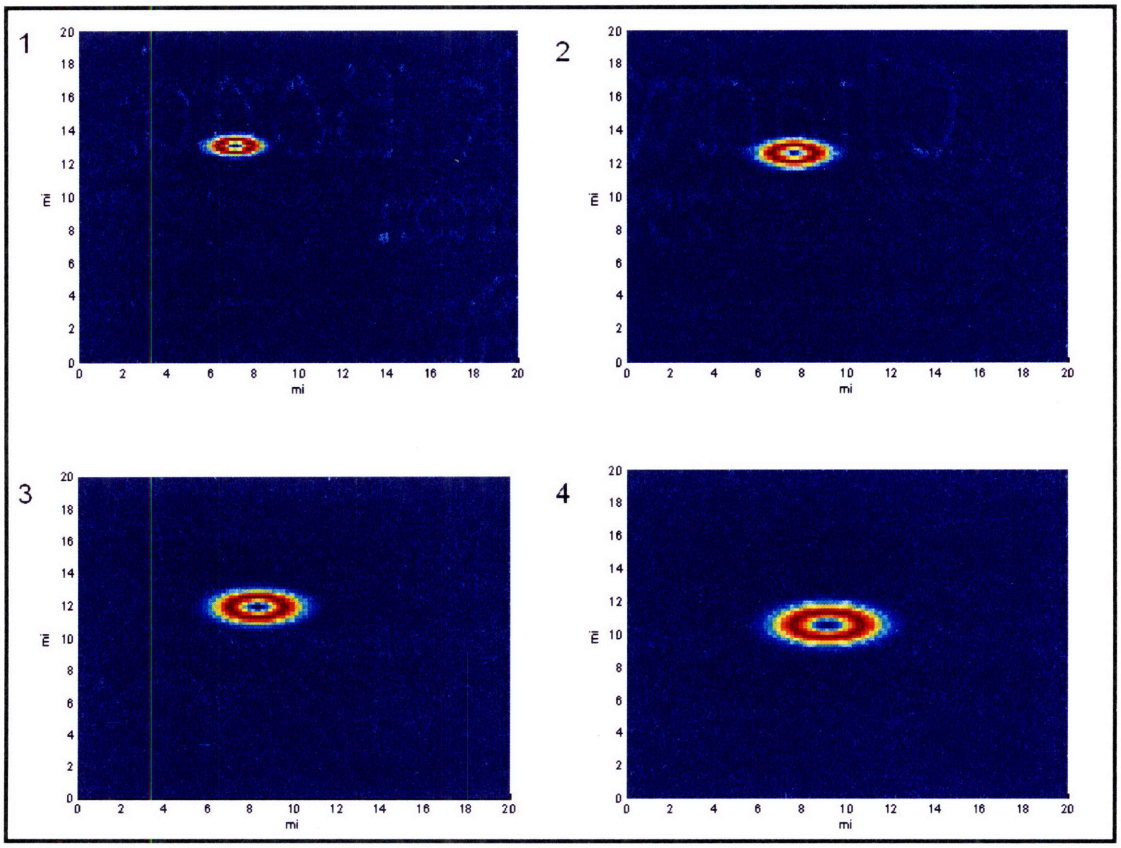



**Figure 5.8: HAB Movement by Day**

This example includes 1052 targets with positive value out of an area having 6561 possible targets. Day 1 includes 112 targets, Day 2 includes 192 targets, Day 3 includes 305 targets, and Day 4 includes 443 targets. We eliminate the other 5509 targets from consideration to limit the size of the problem sent to the algorithms. The optimal plan is not affected by eliminating targets whose value is zero, so this is always done.

We first solved this problem with the greedy algorithm to find the baseline solution. For this example, we utilized only OASIS USVs. Initially, with an observation time at each location of 5 minutes, the greedy algorithm could observe every target because the targets are within a very small geographical region, have large time windows (a full day), and small observation times. Being able to observe every target does not require the algorithm to have to be efficient, so we increased the observation time required at each target until the greedy algorithm could only observe approximately half of the targets. To accomplish this, we had to increase the observation time required at each target to 40 minutes. Figure 5.9 illustrates the routes generated by the greedy algorithm.

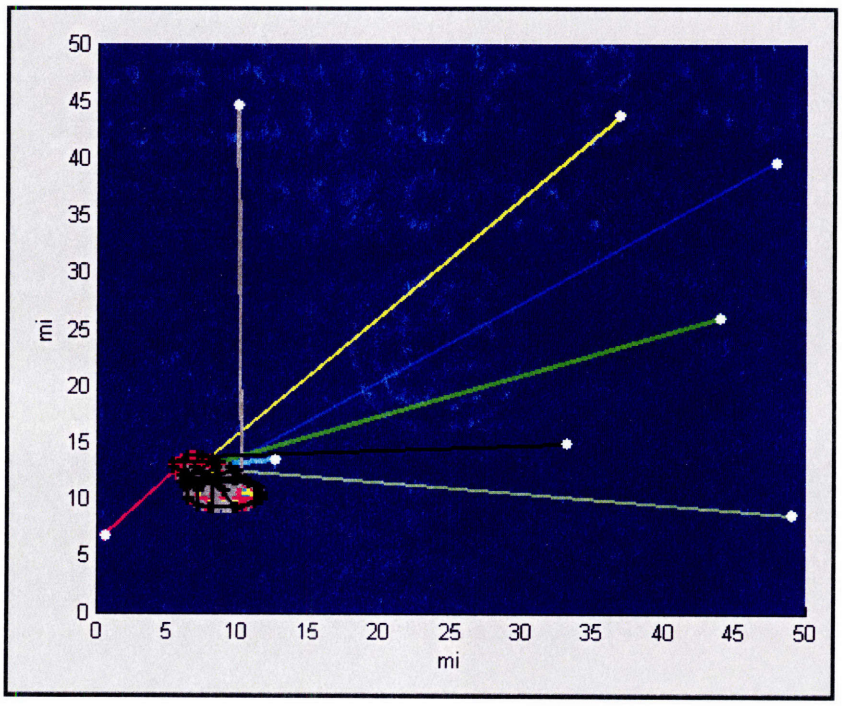



**Figure 5.9: HAB Example Greedy Routes**

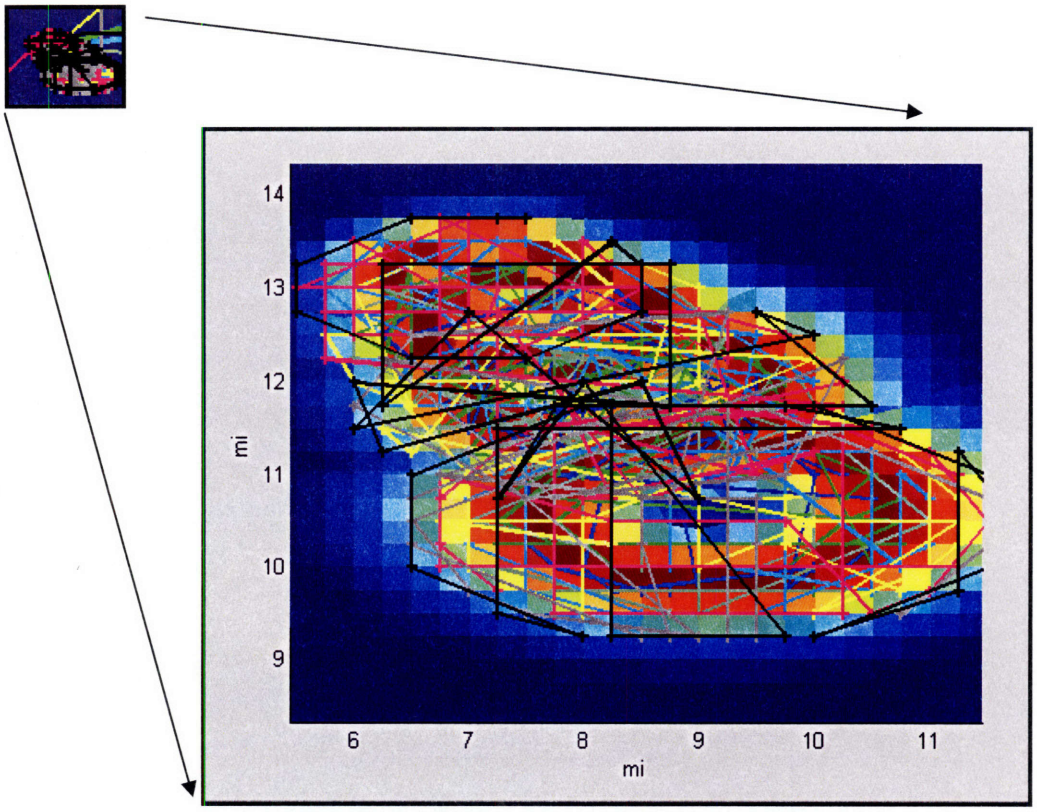



**Figure 5.10: Interior of Example Greedy Routes**

To better illustrate the convoluted results, Figure 5.10 illustrates the convoluted interior region of the HAB. In addition, we show how only one USV is routed within the fleet. Figure 5.11 illustrates the first 24 hours for the 4th USV. Most of its time is spent traveling to the HAB area.

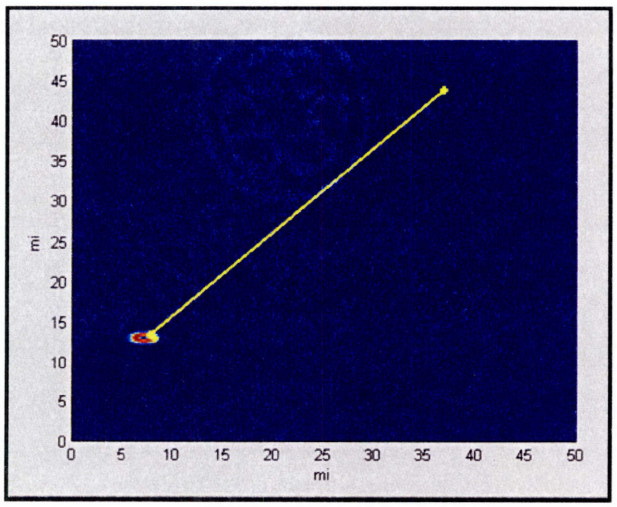



**Figure 5.11: Route 4 Day 1**

Figure 5.12 illustrates the next 72 hours of the 4th USV's route. We changed the color of the route for visual purposes. The greedy algorithm routes the USV in semi-ellipses that visit the higher value targets for each day. The other routes for this example all look similar to Figure 5.12.

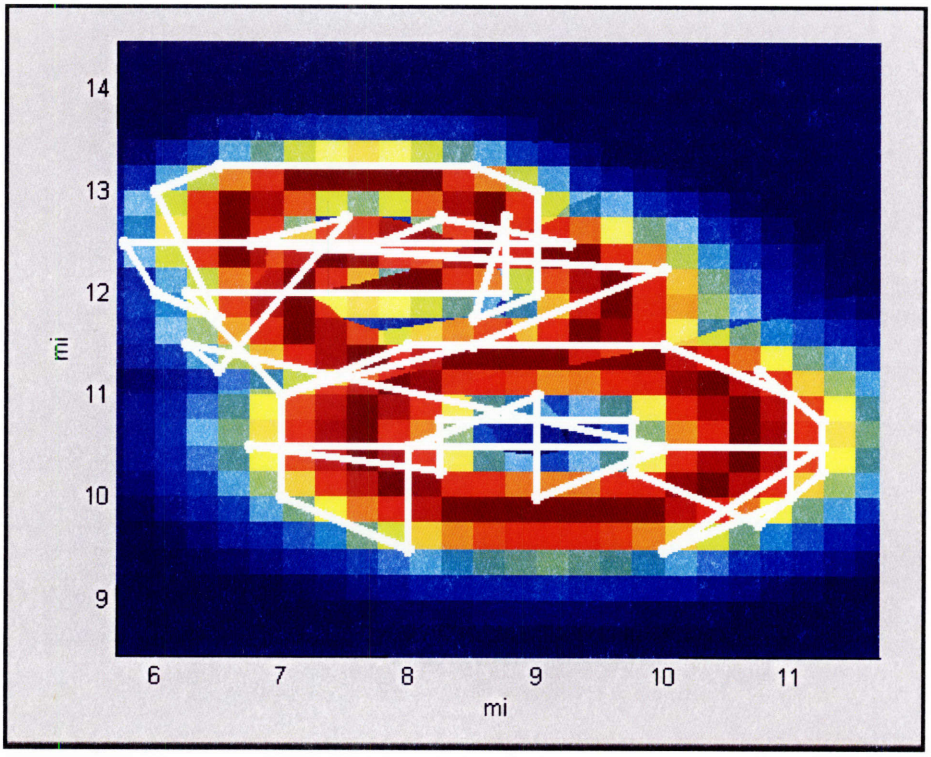



**Figure 5.12: Route 4 Days 2-4**

Next, we used the 3PAA to solve the same dataset. Figure 5.13 illustrates the 3PAA routes. From the figures, we see that the routes for the 3PAA look very similar to the routes constructed by the greedy algorithm. In addition, after considering the results in Table 5.4, we conclude the GCH performed well compared to the 3PAA. The 3PAA inserted an additional 20 targets, but only achieved minimal objective value gains.

| | Greedy | 3PAA |
|---|---|---|
| Number of Targets Served | 459 | 479 |
| Objective Value | 1925.17 | 1939.81 |
| Run Time (s) | 0.9 | 224 |

**Table 5.4: Example HAB Results**

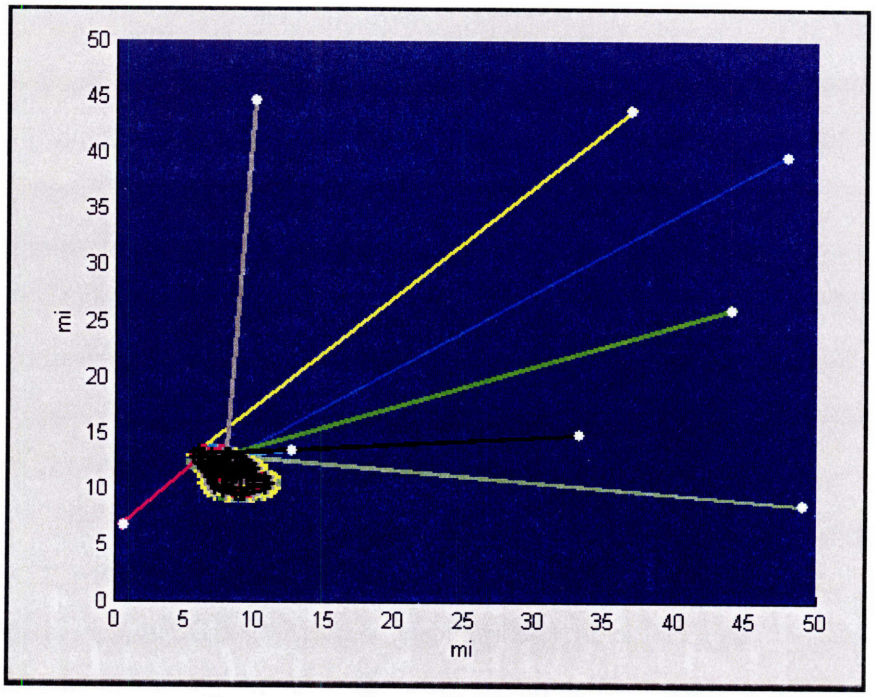



Figure 5.13: 3PAA Routes

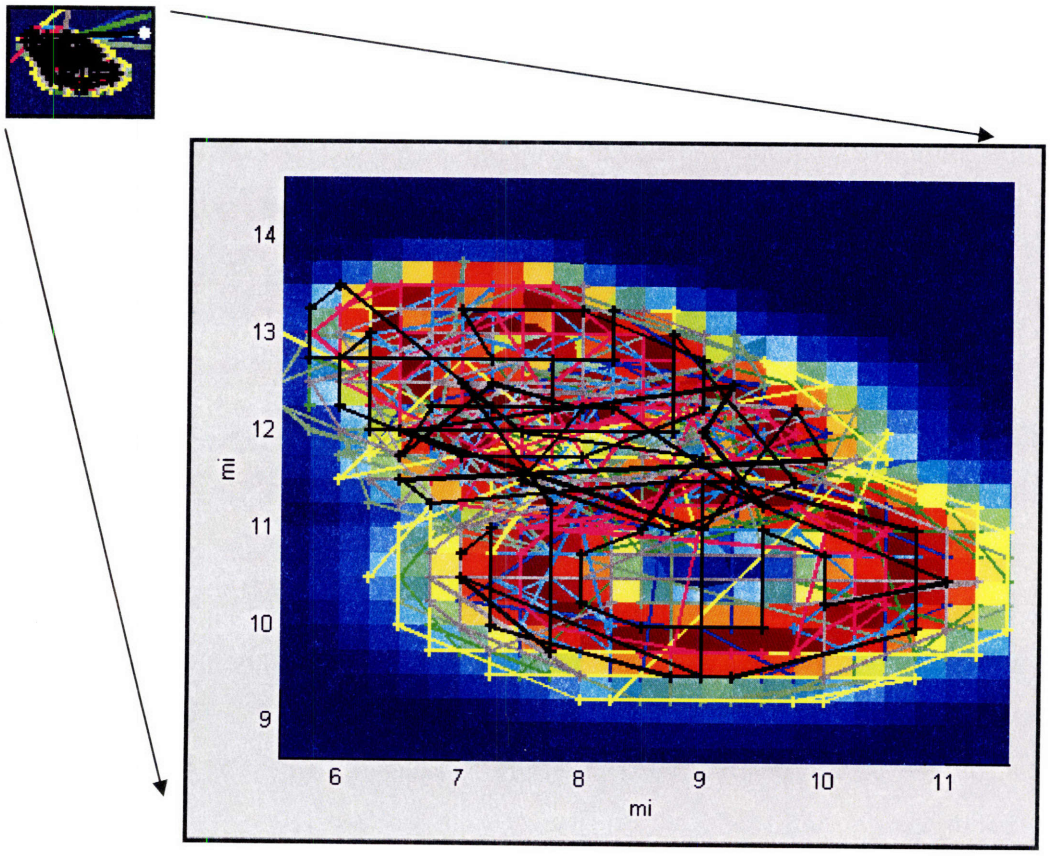



Figure 5.14: 3PAA Interior Routes

Figure 5.14 focuses on the convoluted interior region of the HAB. We found that this example illustrates a situation when the greedy algorithm and the 3PAA provide almost identical results. In cases where the density of targets in relation to the travel time is low, and time windows are large, as in this example, the greedy algorithm can compensate for inefficient routing because ample time exists for the USVs to cross paths. The limiting factor in this example was the high required observation time at each target. To add a target to a route, a route needed at least 40 minutes of slack. The next example illustrates a case when the 3PAA algorithm performs much better than the greedy algorithm.

**HAB Example 2:**

This example changes the first example by decreasing the required observation time at each target from 40 minutes to 5 minutes, and doubling the geographical size of the HAB. However, we cut the density of the targets in order to preserve the number of targets for the problem.

| | Greedy | 3PAA |
|---|---|---|
| Number of Targets Served out of 1052 | 210 | 269 |
| Objective Value | 1102.73 | 1328.29 |
| Run Time (s) | 0.8 | 60.5 |

**Table 5.5: HAB Example 2 Results**

This example illustrates the cases where the 3PAA improves the solution greatly. In this particular example, the 3PAA improved the objective value by 20% while solving in just over one minute.

Through testing many different HAB scenarios, we found several interesting trends:

- In general, HAB scenarios are small geographical areas, and depending on the density of the targets, can usually be fully covered by a fleet of OASIS USVs. For example, 10 USVs can fully cover for densities of 20 targets per 1 square mile per day for HABs sized as in our examples.

- The higher the required observation time, the less the improvement phase and insertion phase matter to the 3PAA. For higher observation times, the algorithm is forced to find larger gaps in the route schedule to fit new targets. Small improvements do not make a difference to the objective function when large gaps are required.
- In general, the most consistent, best performing parameter setting was created when the observation and travel time factors were set to high priority, and time window and wait time factors were set to low priority.
- Spartan Scout USVs can only travel 2.5 times faster than OASIS USVs travel over a 96 hour planning horizon. The extra speed improves the objective value solutions; mainly due to the fact that they respond faster to the region of interest. However, their improvement is lessened when required observation times are high, the HAB is small, the length of scenario duration increases, the USVs start closer to the region of interest, or the number of targets is small. In fact, if the Spartan Scout is to be active for a whole week, then it has to constrain its speed to equal the OASIS speed. In addition, OASIS USVs are the only option for scenarios lasting longer than one week. Usually, a fleet of OASIS USVs provide enough speed to cover the HAB scenarios for problems with 8 or more USVs, HABs that spread less than 15 miles across lengthwise at determination, and the density is less than 20 targets per one square mile.

### 5.4.4 Hurricane Scenario

Now, we explore the results of the 3PAA on the realistically sized hurricane datasets. For the following examples, we tested datasets where the number of targets ranged from 500 to 3000 targets. We chose this range for the number of targets to explore because changes in the algorithm in this range dramatically change the run time and solutions. For smaller problems, changes in the algorithm only have minor effects on the results. We describe how the algorithm performs through two examples.

**Hurricane Example 1:**

Initialization:

Figure 5.15 illustrates a hurricane scenario where a hurricane has been identified and 8 USVs in the area are called to take measurements ahead of the storm. The white dots represent the initial

USV locations and the hurricane is currently located in the bottom-left of the region of interest. The color indicates the initial value of observations. The hurricane is predicted to move at 15 knots across the grid, the latest time window in the upper-right of the grid is approximately 9.4 hours. Therefore, the planning horizon for this scenario is 9.4 hours, which is the time the hurricane would move across the grid. Figure 5.16 illustrates the late times of each of the targets. The blue color represents the earlier deadlines, and the red represents the later deadlines, which correspond to the projected hurricane's speed across the grid.

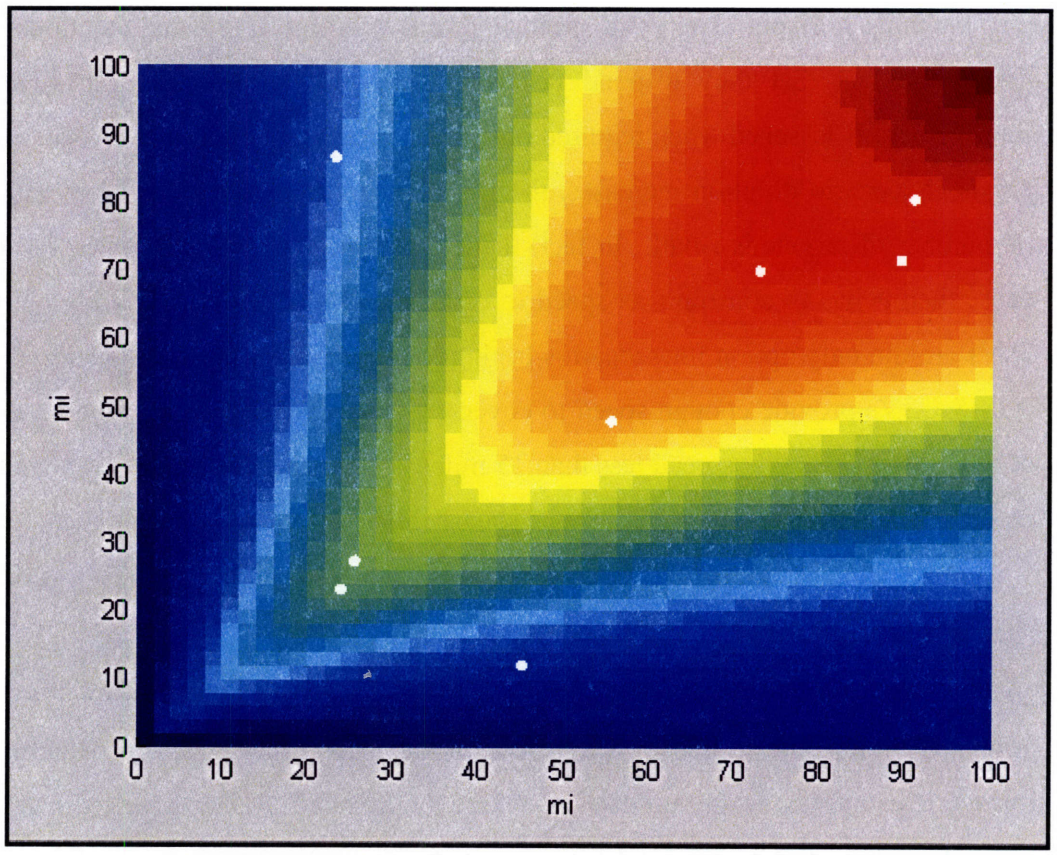



**Figure 5.15: Initial Hurricane Target Values and USV Locations**

This example includes 2601 targets with positive value. We first solved this problem with the greedy algorithm to find the baseline solution. For this example, we utilized only OASIS USVs. We set the required observation time at each target to zero, because the OASIS USVs move slowly enough that they can collect the temperature reading as they move to the next target. Figure 5.17 illustrates the routes generated by the greedy algorithm.

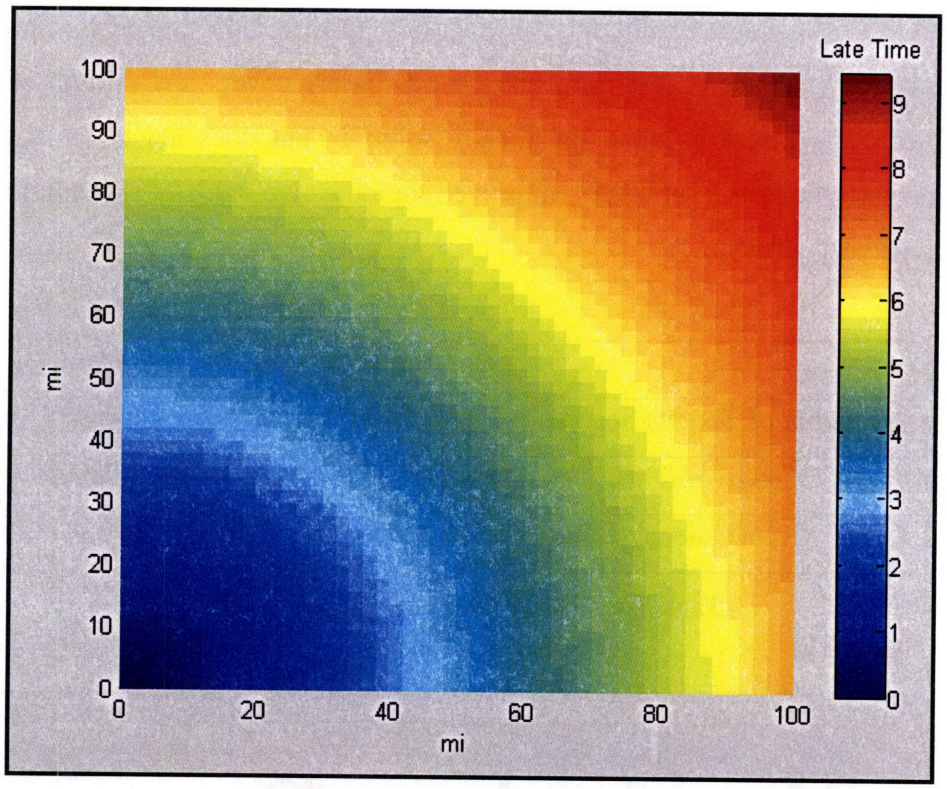



**Figure 5.16: Target Deadlines**

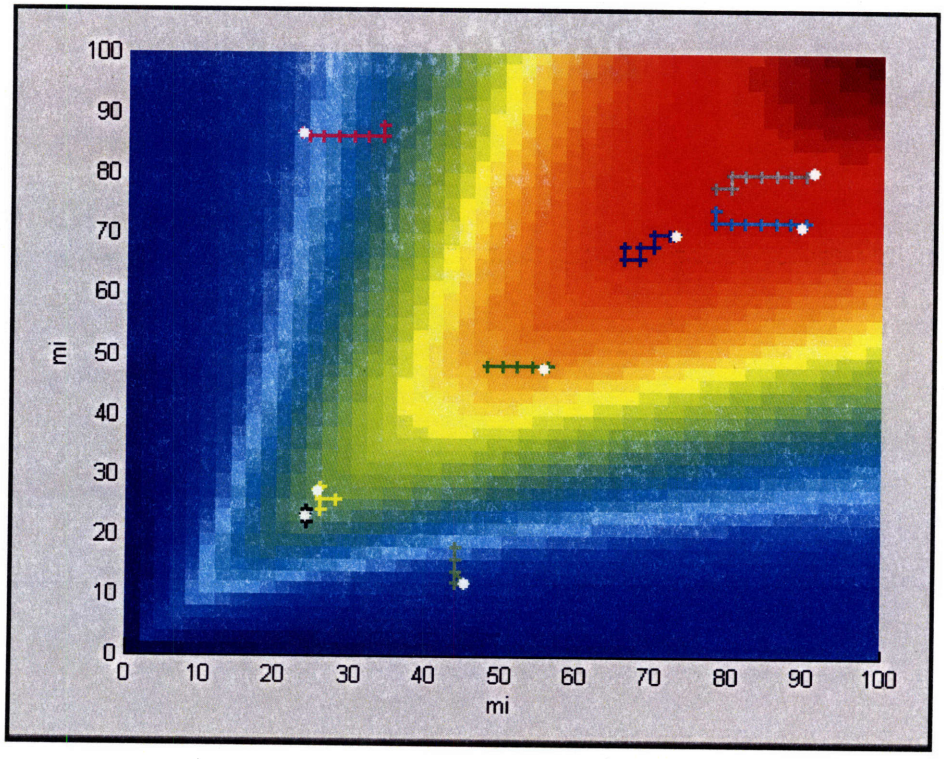



**Figure 5.17: Greedy Routes OASIS**

Figure 5.17 illustrates the ineffectiveness of the OASIS USV for the hurricane scenario. With a larger area to cover in less time, the OASIS barely covers any portion of the area of interest. The 3PAA improves upon the greedy solution, but does not cover any significant portion of the entire region. The 3PAA improved the objective value of the solution by almost 14% and solved very quickly.

| | Greedy | 3PAA |
|---|---|---|
| Number of Targets Served | 43 | 49 |
| Objective Value | 278.36 | 328.03 |
| Run Time (s) | 0.4 | 49 |

**Table 5.6: Example Hurricane Results OASIS**

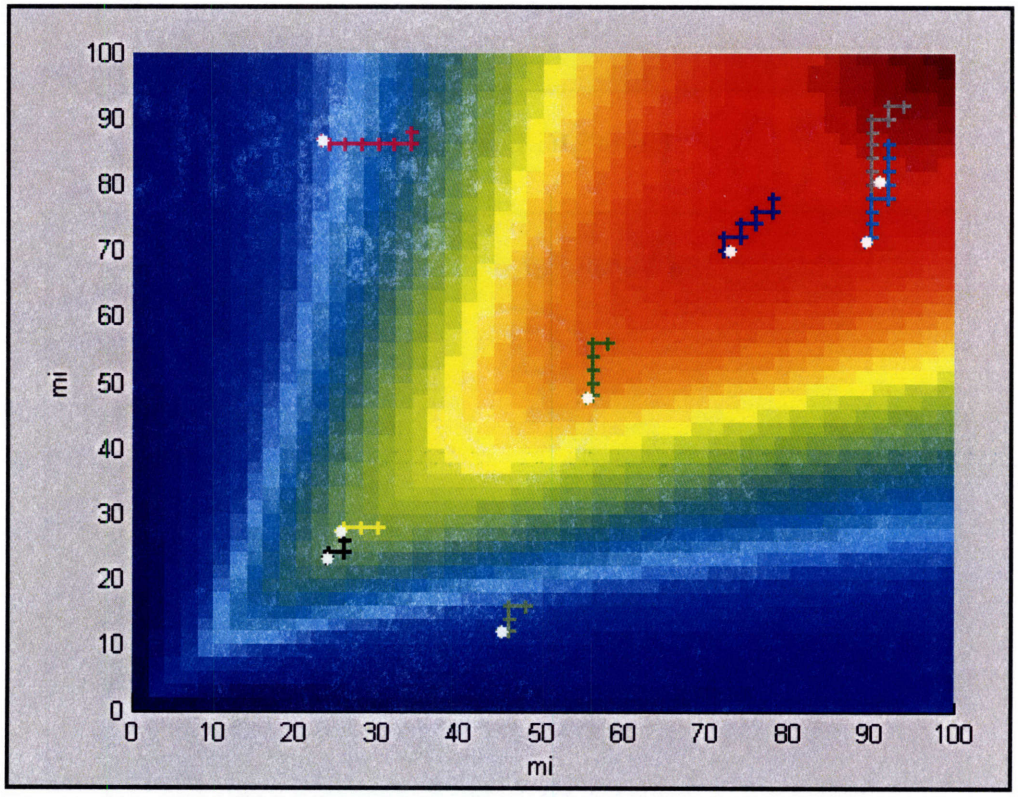



**Figure 5.18: 3PAA Routes OASIS**

Figure 5.18 illustrates the 3PAA routes. Note that 3PAA selects routes that are rather different from the routes selected by GCH. The 3PAA utilizes a parameter setting where $w3 = 1$,

which collects the highest available value as quickly as possible because travel time is the only priority, which is the main difference from the 3PAA and the greedy algorithm. The greedy algorithm balances between the different factors and steers the USVs towards targets with earlier time windows before it steers them to the high value targets.

**Hurricane Example 2:**

This example changes the first example by utilizing the Spartan Scout USV instead of the OASIS USV. The speed of the Spartan Scout is 8 times faster than the OASIS platform over a 10-hour scenario. This allows for observation of many additional targets in the area of interest. Figure 5.19 illustrates the greedy algorithm's routes that generally initially head towards the most likely path of the hurricane. Then the routes travel towards the hurricane's projected path to observe the early time window targets. Then the algorithm turns the USVs 180 degrees around and heads them toward the high value targets with later time windows.

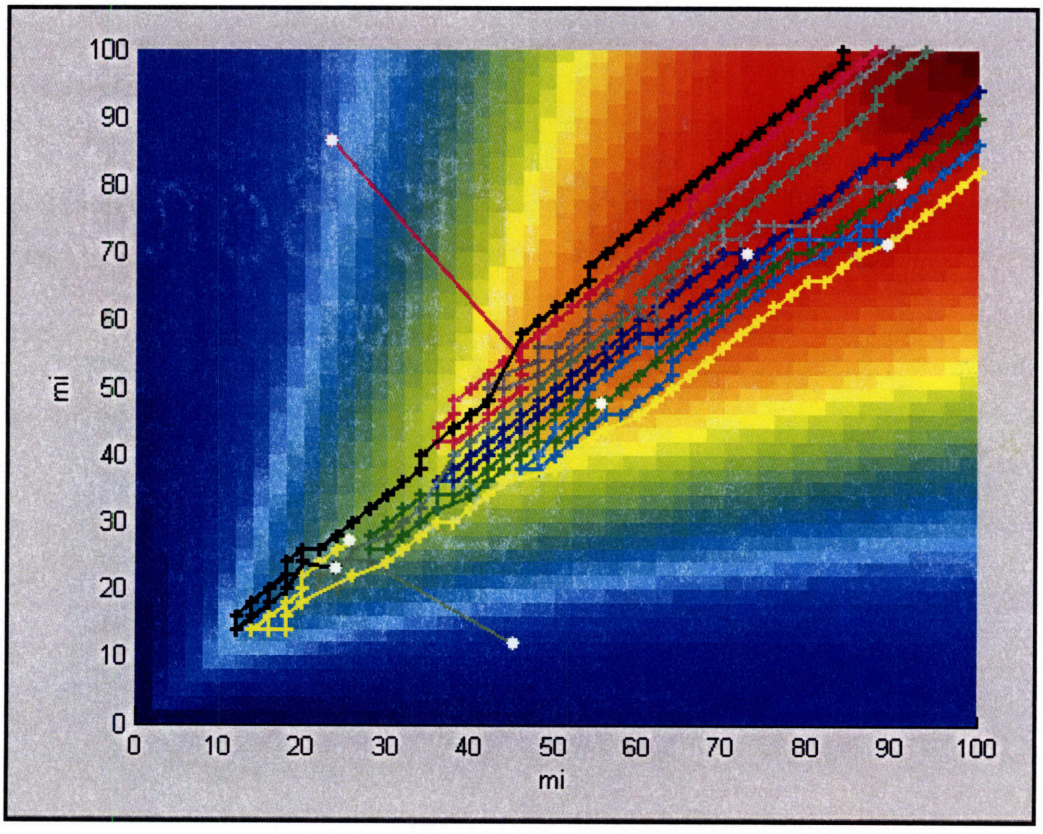



**Figure 5.19: Greedy Routes Spartan Scout**

The 3PAA improved the objective value of the solution by over 34% from the greedy algorithm. However, the 3PAA processed slower because more targets were feasible. Figure 5.20 illustrates the routes that the 3PAA constructed. The Spartan Scout USV collected 1000% more value than the value collected by the OASIS USV. In addition, the Spartan Scout USV is more durable and feasibly a better option to survive a hostile hurricane environment. In addition, it can travel to stay ahead of the storm, but an OASIS USV cannot.

| | Greedy | 3PAA |
|---|---|---|
| Number of Targets Served | 363 | 448 |
| Objective Value | 2542.98 | 3412.78 |
| Run Time (s) | 0.4 | 556 |

**Table 5.7: Hurricane Example 2 Results**

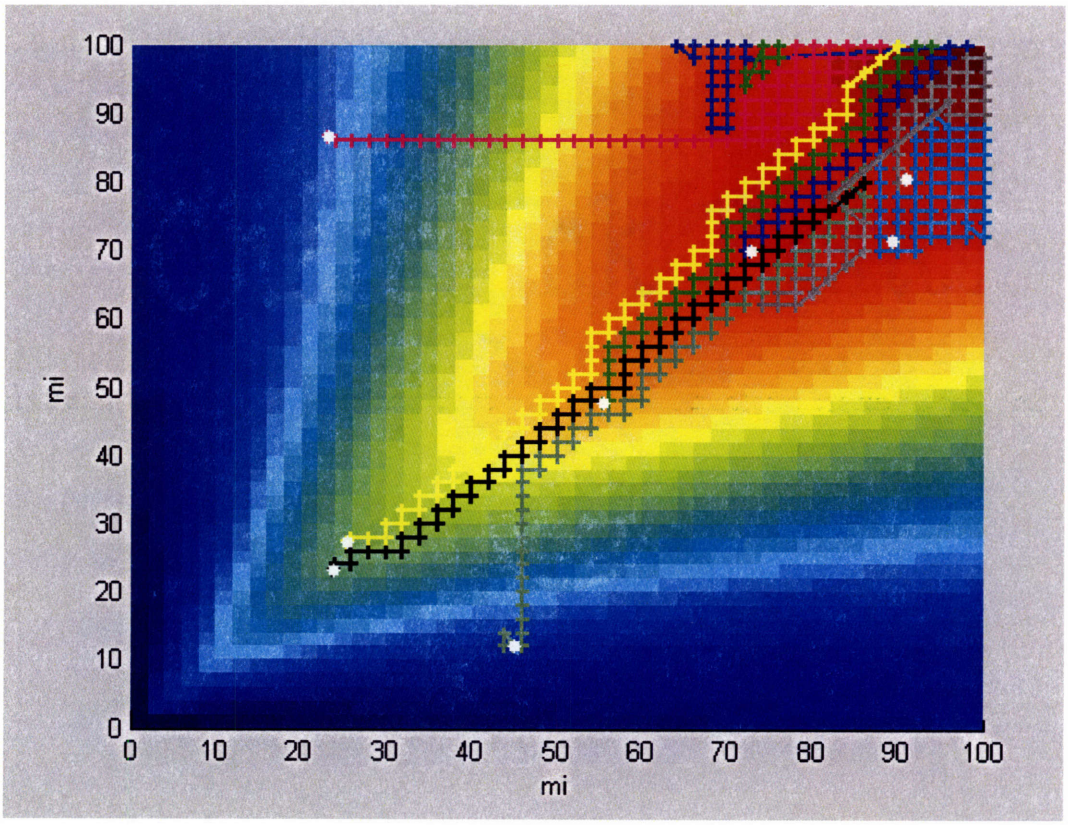



**Figure 5.20: Hurricane Example 3PAA Spartan Scout**

The routes that produced the best solution for the 3PAA were initialized with the travel time parameter as the only priority. This produced routes that observed targets as close together as possible, while following the gradient that increased the objective function the fastest. It remains the best parameter setting even if we lengthen the scenario, add additional USVs, or decrease the number of targets.

Through testing many different hurricane scenarios, we found several interesting trends:

- In general, hurricane scenarios are very large geographical areas, and depending on the density of the targets, cannot usually be covered even moderately by a fleet of OASIS USVs. OASIS USVs produce ineffective results for these scenarios because hurricanes move so much faster than they do.
- In general, the most consistent, best performing parameter setting was created when the time window and travel time factors were set to high priority, and observation and wait times factors were set to low priority. However, for large datasets and less USVs, the best setting was created by using only the travel time factor.
- On average, Spartan Scout USVs travel 9 times faster than OASIS USVs for a typical 10-hour scenario, and dramatically improve the objective value solutions by an average of 910% over 10 trials.

## 5.5 Summary

The exact algorithm, greedy construction heuristic, and the 3PAA have been tested and the results reported in this chapter. The significant results:

- The exact algorithm in ILOG solved cases of up to 40 targets. However, even for cases with 25 targets, it took several hours to solve.
- On average, we found the 3PAA solves within 5% of optimal in the pseudo-random cases that the exact algorithm could solve.
- On average, we found the 3PAA improves the greedy construction heuristic by 77% over the pseudo-random datasets, without a significant increase in run-time performance.
- On average, HAB scenarios can utilize OASIS USVs without much drop-off from the Spartan Scout USVs. In addition, if the scenario is longer than a week, then the OASIS's endurance makes it the better option.

- The Spartan Scout USV is the only option in the hurricane scenario. The OASIS USV travels too slowly to monitor a significant number of targets.
- The 3PAA can be tweaked to solve each scenario faster.
    a. For the HAB scenario, we set the observation and travel time factors to high priority, and set the time window and wait time factors to low priority. In order to increase the run-time performance, we limit the 3PAA to consider only this parameter combination to cut the run-time by an average of 85% compared to the standard settings without significant effect on the final objective function value.
    b. For the hurricane scenario, we also set the algorithm to run for only one parameter combination. We set the travel time as the only factor. This cuts the average run-time for each scenario by 85% compared to the standard settings without significant effect on the final objective function value.
- On average, the 3PAA improves the greedy heuristic by approximately 10% in HAB scenarios with 1000 targets with an increase in solve run-time of only one minute.
- On average, the 3PAA improves the greedy heuristic by approximately 20% in hurricane scenarios with 2600 targets, but with an increase in solve run-time in less than 8 minutes.

# Chapter 6

# Conclusions and Future Work

This chapter summarizes the contributions of this thesis and presents suggestions for future research related to this thesis. We present general comments concerning the test results, our formulations, and the heuristic algorithms. We make suggestions on how the heuristics might be improved. Finally, we provide recommendations for future work in the field.

## 6.1 Thesis Contributions

The goal of this research was to develop an automated observation planner for a fleet of USVs that allocates the fleet efficiently and takes advantage of the USV's true potential. We developed an algorithm that solved the USVOPP in realistically sized problems in a reasonable amount of time. In every case, up to 3000 targets and 24 USVs, we solved the USVOPP in less than 30 minutes.

This research makes the following contributions:

1. An MIP formulation for the USVOPP
2. The development and implementation of the optimal formulation implemented in ILOG OPL Studio 5.0 using CPLEX v10.1.0 that could solve USVOPP test cases of up to 40 targets in 36 hours
3. The development and implementation of a new greedy construction heuristic to solve the USVOPP implemented in Java to serve as a baseline to test against that solved problems as large as 16,275 targets in less than 10 minutes
4. The development and implementation of a new three-phase heuristic algorithm to solve the USVOPP implemented in Java that, on average, solved problems with 2600 targets in less than 8 minutes and for problems with 10,000 targets 17 hours and 18 minutes
5. The development of a series of test datasets for hurricane and harmful algal bloom scenarios for the USVOPP
6. Experimentation and comparison of the implemented algorithms
7. Recommendations for future modifications and future work related to solving the USVOPP and similar problems in general

## 6.2 Recommendations on Modification

Through the development and testing of the 3PAA, we found possibilities to change the USVOPP formulation in both its application and its composition. In addition, we comment on general modifications that could help the 3PAA.

### 6.2.1 Problem Formulation Modifications

We could allow targets to be observed more than once. In our formulation, we allow targets at the same geographical location, but they are otherwise independent. It could be valuable to calibrate a target by observing it with two separate USVs. Incorporating this modification would add considerable complexity to the problem, as it would involve the use of a

separate decision variable for every subsequent observation of a target. However, it would allow more flexibility in the types of scenarios that can be modeled.

In addition, we could model the possibility that USVs are lost or damaged in the course of operations. USVs could be lost or damaged as a result of a vessel malfunction, loss of power, or the natural environment. In our formulation, we could solve this issue with a re-plan. However, this possibility could be specifically planned for by the formulation. It would be good to have a route structure that is suboptimal, but adds slack in the model so that small changes (i.e. a USV breakdown) do not disrupt the whole plan. This is called robust planning. Planning with robustness built into the observation-planning model could provide better results in practice.

Another modification to the USVOPP would be to include the tidal currents into the calculation of travel time between targets. This adds much complexity to the model as each USV would need its heading to calculate how much time is subtracted or added. In addition, different geographical areas have different currents, so the calculation is time and location dependent. In addition, tidal currents that can be as fast as 1.3 knots can have great consequences on the observation planning for OASIS USVs, which only travel at two knots.

Another modification to the USVOPP formulation would be to include coordination between multiple planning periods. Throughout this thesis, we assumed that observations took place during a single planning period without any possible pertinence to the previous or next planning periods. A more sophisticated situation emerges when the observations need to be carried out over multiple planning periods. The scenarios we covered included the region of interest that is dynamic and transforming in shape as time progresses, and the USVs moved with the moving field to collect as much useful data as possible. In scenarios like this, it is necessary to have some information exchange and coordination between the paths of the vessels that must be realized on consecutive planning periods to satisfy path optimality over time.

To demonstrate, one simple approach to observing over multiple planning periods could be to treat each planning period independently, as we did in this thesis. Consider the HAB scenario. For the planning period, a USV has a 4-day route to follow. However, after the 4-day route, there is a significant probability that the HAB still exists. The vessel starts its motion for the following planning period from where it finishes the previous planning period. This case can be solved by splitting the problem into two single-planning scenarios and treating each of them independently. The problem is that there is no coordination between the different periods, and

although the solution obtained will be spatially optimal, there is no guarantee that it will also be optimal over time. This approach is myopic because it does not look forward in time. For instance, on the first planning period the vessel might head to a high-value region that lies in one area of the map. However, if the high value region for the second period develops in a distant area, this region now becomes out of the range of the vessel. Instead, the vessel will spend its time observing a lower-value region. This behavior could also extend to the third planning period as well, since there will still be no exchange of information between the second and the third planning periods.

An alternative approach is to incorporate all the available probable value data for consecutive periods at once into the formulation and solve all of the routes concurrently for consecutive planning periods. The value data for the routes during the first period become available for the second; conversely, the value data for the routes to be traveled during the second period becomes available for the first planning period. However, this does not mean that one set of the routes for either period is constructed first and the other set of routes is constructed subsequently. Instead, both route sets for each planning period are solved simultaneously as a function of each other. By maintaining this two-way communication link, time global optimality is also satisfied. Our approach lacked the necessary implementation flexibility to handle this time-progressive case, since the starting points of the vessels for each period must be known beforehand. However, this approach quickly leads to problem instances that become very large.

### 6.2.2 3PAA Modifications

In regards to the 3PAA algorithm, one modification includes randomizing the order for the improvement phase. We could complete inter-route improvements first and intra-route improvements second. In addition, we could also change the phase order to complete maximal insertions first, and then complete the improvement phase. Just as we experimented with the different combinations of the parameters, we could experiment with the different orders of the improvement methods and phases.

Another interesting modification would be to calculate the ratios of the target value to the four factors, as described in Section 5.2.2, in the construction algorithm and the insertion algorithm with a 1-step look-ahead method. For example, we currently calculate the ratio based

on how the next target improves the current route. However, if we calculated the ratio based on how that target's selection affects the next insertion, we would avoid sending USVs to observe targets that are the next best available target, but cause the USVs subsequent moves to be poor. This method causes a USV to look constantly for the best move that also positions it for the next best move. However, this method would add complexity and run-time to the algorithm.

Additional modifications to the 3PAA would be to test an insertion-deletion heuristic, a 3-opt heuristic, an Or-opt heuristic, or a cyclic exchange heuristic. Our focus was to keep run-time low, but in order to produce good solutions; all of these heuristics would be good options to explore adding to the algorithm. It would be good to implement any of these heuristics when the algorithm cannot improve with other methods, or for smaller problems that would not affect the run-time performance.

## 6.3 Future Work

There are several opportunities for future research in addition to the modifications suggested in the previous section. The first area involves solving the USVOPP with any of the other methods we briefly mentioned in this thesis. In future work, this research could be continued by examining how an Ant Colony System Algorithm, a Simulated Annealing Algorithm, a Genetic Algorithm, a Neighborhood Search, or a Tabu Search Algorithm compares to the local search algorithm we used to solve the problem. Any of these methods could be better or worse than our algorithm.

In addition, another area of future work could be to apply our USVOPP formulation to different problem areas. Scheduling problems involving UAVs, cargo aircraft, or trucks admit similar formulations. A current problem is to observe wildfires in the western United States with UAVs. Sensor webs that provide coverage of wildfires are currently being implemented that will link satellites, UAVs, and ground-based sensors in a sequence of analysis activities that deliver information required by management analysts in response to national, regional, or local emergencies [6], [7]. Our formulation for the USVOPP can be modified to solve this problem.

**[This Page Intentionally Left Blank]**

# Appendix A: Glossary of Acronyms

| | |
|---|---|
| 3PAA | Three-Phase Approximate Algorithm |
| AMSR-E | Advanced Microwave Scanning Radiometer |
| ASF | Adaptive Sensor Fleet |
| C-MAN | Coastal-Marine Automated Network |
| EOS | Earth Observing System |
| GA | Genetic Algorithm |
| GCH | Greedy Construction Heuristic |
| GIS | Global Information Systems |
| GPS | Global Positioning System |
| HAB | Harmful Algal Bloom |
| IDE | Integrated Development Environment |
| MIP | Mixed Integer Programming |
| MODIS | Moderate-resolution Imaging Spectroradiometer |
| NASA | National Aeronautics and Space Administration |
| NHC | National Hurricane Center |
| NOAA | National Oceanic & Atmospheric Administration |
| NP | Nondeterministic Polynomial-time |
| NWS | National Weather Service |
| OASIS | Ocean-Atmosphere Sensor Integration System |
| OP | Orienteering Problem |
| OPTW | Orienteering Problem with Time Windows |
| PCTSP | Prize Collecting Traveling Salesman Problem |
| PDF | Probability Density Function |
| PPFSS | Prototype Phytoplankton Fluorescence Sensing System |
| SBS | Space-Based Sensor |
| TOP | Team Orienteering Problem |
| TOPTW | Team Orienteering Problem with Time Windows |
| TSP | Traveling Salesman Problem |

| | |
|---|---|
| TSPTW | Traveling Salesman Problem with Time Windows |
| UAV | Unmanned Aerial Vehicle |
| UHF | Ultra High Frequency |
| USV | Unmanned Surface Vessel |
| USVOPP | Unmanned Surface Vessel Observation-Planning Problem |
| VHF | Very High Frequency |

# References


[1] "A Plan for Co-ordinated Scientific Research and Co-operation to Develop International Capabilities for Assessment, Prediction and Mitigation," GEOHAB: Global Ecology and Oceanography of Harmful Algal Blooms. Joint IOC / SCOR Workshop Report, Oct. 1998.

[2] Aarts, E., and Lenstra, J. K. "Local Search in Combinatorial Optimization," John Wiley & Sons, New York, 1997.

[3] "Adaptive Path Planning, Map Building and Situation Awareness to Support ISR Operations." The Charles Stark Draper Laboratory, Inc. Technical Proposal, 19 July 2001.

[4] Ahuja, R. K., Ergun, O., and Orlin, J. B., "A survey of very large-scale neighborhood search techniques," *Discrete Applied Mathematics,* 123:75-102, 2002.

[5] Ahuja, R. K., Orlin, J. B., Sharma, D., "Very Large-Scale Neighborhood Search," *International Transactions in Operations Research 7*, 301-317, 2000.

[6] Ambrosia, Vincent G., Brass, James A., Greenfield, Paul, and Wegener, Steve, "Collaborative Efforts in R&D and Applications of Imaging Wildfires," *US Forest Service RS2004 Conference Paper,* 2004.

[7] Ambrosia, V.G., S.W. Buechel, J.A. Brass, J.R. Peterson, R.H. Davies, R.J. Kane and S. Spain. "An Integration of Remote Sensing, GIS, and Information Distribution for Wildfire Detection and Management," *Photogrammetric Engineering and Remote Sensing*, 64(10): 977-985, 1998.

[8] Archetti, Claudie, Hertz, Alain, and Speranza, Maria Grazia, "Metaheuristics for the team orienteering problem," *Journal of Heuristics* 13(1):49-76, 2007.

[9] Balas, E., Martin, G., Roll-A-Round, *Software Package for Scheduling the Sounds of a Rolling Mill*, Copyright Balas and Martin Associates, 104 Maple Heights Road, Pittsburgh, 1985.

[10] Balas, E., "The Prize Collecting Traveling Salesman Problem," *Networks* 19, 621-636. 1989.

[11] Battiti, R., "Reactive search: Toward self-tuning Heuristics," In V. J. Rayward-Smith, editor, *Modern Heuristic Search Methods*, chapter 4, pages 61--83. John Wiley and Sons Ltd, 1996.

[12] Bertsekas, D. P., *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, 2000.

[13] Bodin, L., Golden, B., Assad, A., and Ball, M., "Routing and schedule of vehicles and crews: The state of the art." *Computational Operations Research*, 62-212., 1983.

[14] Bullnheimer B., Hartl, R.F., and Strauss, C., "Applying the Ant System to the Vehicle Routing Problem." Department of Management Science, University of Vienna. 1997.

[15] Bushaw-Newton, K.L. and Sellner, K.G. (on-line), "Harmful Algal Blooms. In: NOAA's State of the Coast Report." Silver Spring, MD: National Oceanic and Atmospheric Administration, 1999 <http://state-of-thecost.noaa.gov/bulletins/html/hab_14/hab.html>

[16] Butt, Steven E. and Ryan, David M. "An optimal solution procedure for the multiple tour maximum collection problem using column generation," *Computers & Operations Research*, 26 427-441, 1999.

[17] Chao, I. M., Golden, B. L., and Wasil, E. A., "A Fast and Effective Heuristic for the Orienteering Problem," *European Journal of Operational Research* 88, 475-489, 1996.

[18] "Coastal Observation II – A Continuation of the OASIS Project" NOAA Semi-annual Technical Report, Feb. 2005.

[19] Colorni A., Dorigo, M., and Maniezzo, V., "Distributed Optimization by Ant Colonies." *Proceedings of ECAL91 - European Conference on Artificial Life*, Paris, France, F.Varela and P.Bourgine (Eds.), Elsevier Publishing, 134–142. 1991.

[20] Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., Soumis, F. "VRP with Time Windows". In P. Toth and D. Vigo (eds.): "The Vehicle Routing Problem," SIAM Monographs on Discrete Mathematics and Applications, vol. 9, Philadelphia, PA, 157-193, 2002.

[21] Dantzig, G. B., Fulkerson, R., and Johnson, S. M., "Solution of A Large-Scale Traveling Salesman Problem," *Operations Research 2*, 393-410, 1954.

[22] de Farias, D. P., "The Linear Programming Approach to Approximate Dynamic Programming: Theory and Application," Dissertation, Stanford University, June 2002.

[23] Emanuel, Kerry. "Anthropogenic Effect on Tropical Cyclone Activity," Jan. 2006 <http://wind.mit.edu/~emanuel/anthro2.htm>

[24] "Estimated Annual Economic Impacts from Harmful Algal Blooms (HABs) in the United States," Technical Report, Sep. 2000. <http://www.whoi.edu/redtide/pertinentinfo/Economics_report.pdf>

[25] Fischetti, M., and Toth, P., "An Additive Approach for the Optimal Solution of the Prize-Collecting Traveling Salesman Problem,". In *Vehicle Routing: Methods and Studies*, Golden, B.L., Assad, A.A., eds., North Holland, Amsterdam, 1988.

[26] Fischetti, M., Gonzalez, J. J. S., and Toth, P., "A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem", *Working Paper*, University of Bologna (to appear in Operations Research), 1994.

[27] Gendreau, M., Laporte, G., and Stan, M. "A Generalized Insertion Heuristic for the Traveling Salesman with Time Windows," *Operations Research* 43, 330- 335, 1998.

[28] Glover, F. "Tabu Search — Part I," *ORSA Journal on Computing 1*: 3, 190-206, 1989.

[29] Glover, F. "Tabu Search — Part II," *ORSA Journal on Computing 2*: 1, 4-32, 1990.

[30] Gordon, Neil and David Salmond. "Sequential Monte Carlo Methods for Tracking." UK Adaptive Signal Processing Club Discussion Meeting, 7 Mar. 2001. <http://www.aspc.qinetiq.com/>.

[31] Golden, B. L., Levy, L. and Vohra, R., "The Orienteering Problem." *Naval Research Logistics*, Vol. 34, Issue 3, Pages 307-318, June 1987

[32] Golden, B. L., Wang, Q., and Liu, L., "A Multifaceted Heuristic for the Orienteering Problem," *Naval Research Logistics* 35 359-366, 1988.

[33] "Harmful Algal Booms in US Waters,", Report of the National Science and Technology Council Committee on Environment and Natural Resources. Oct. 2000. <http://www.habhrca.noaa.gov/FinalHABreport.pdf>

[34] Holder, Pat. "Unmanned Surface Vehicle (USV) For Assured Access and Force Protection," Department Process Modernization Office SPARTAN Scout ACTD, Feb. 2003.

[35] Holland, John H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[36] http://www.ilog.com

[37] Kantor, Marisa G. and Rosenwein, Moshe B. "The Orienteering Problem with Time Windows." *The Journal of the Operational Research Society*, Vol. 43, No. 6.) pp. 629-635, June, 1992.

[38] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, Vol 220, Number 4598, pages 671-680, 1983.

[39] Laporte, G., "The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms," *European Journal of Operational Research 59*, 345-358, 1992

[40] Laporte, G. and Martello S. "The selective travelling salesman Problem." *Discrete Applied Mathematics*, Vol. 26. pp. 193-207, 1990.

[41] Lin, S., "Computer Solutions to the Traveling Salesman Problem," *Bell System Technical Journal*, 1965.

[42] Lin, S., and Kernighan, B., "An effective heuristic algorithm for the traveling salesman problem," *Operations Research* 21 498-516, 1973.

[43] McAlary, David. "Hurricane Prediction More Reliable Than Ever, But Still Imprecise," VOA News, Sep. 2005. <http://www.voanews.com/english/archive/2005-09/2005-09-21-voa79.cfm>

[44] Mingozzi, A., Bianco, L., Ricciadelli, S., "Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints," *Operations Research* 45, 365-377, 1997.

[45] "NOAA's Hurricane Hunter Aircraft Description. Lockheed WP-3D Orions (P-3s) and Gulfstream IV SP (G-IV) Jet," NOAA. Mar. 2003. <http://www.publicaffairs.noaa.gov/grounders/hurricanehunters.html>

[46] Nordhaus, William D. "The Economics of Hurricanes in the United States," Technical Report, Dec. 2006 <http://www.econ.yale.edu/~nordhaus/homepage/hurr_122106a.pdf>.

[47] Ramesh, R., and Brown, K. M., "An Efficient Four-Phase Heuristic for the Generalized Orienteering Problem," *Computers and Operations Research* 18, 151-165, 1991.

[48] Potvin, Jean-Yves and Rousseau, Jean-Marc, "An Exchange Heuristic for Routeing Problems with Time Windows," *The Journal of the Operational Research Society*, Vol. 46, No. 12, pp. 1433-1446, Dec. 1995.

[49] "Report to the Florida Harmful Algal Bloom Taskforce: Blue Green Algae, Their Toxins & Public Health Issues," Oct. 2001. <http://www.rsmas.miami.edu/groups/niehs/science/pdf/bluegreen2001litreview.pdf>

[50] Rinooy, Kan, A. H. G., Lawler , E. L., Lenstra, J. K., and Shmoys, D. B., "*The Traveling Salesman Problem*," John Wiley, New York, 1985.

[51] Ropke, S., Cordeau, J.-F., and Laporte, G., "Models and a Branch-and-Cut Algorithm for Pickup and Delivery Problems with Time Windows," University of Copenhagen, July 2005.

[52] Rosenkrantz, D., Stearns, R., and Lewis, P. M., "An Analysis of Several Heuristics for the Traveling Salesman Problem." *SIAM J. Computing* 6, 563-581. 1977.

[53] Sample, Sharron. "The Physical Ocean," NASA, June 2005. <http://science.hq.nasa.gov/oceans/physical/index.html>

[54] Sample, Sharron. "Sea Surface Temperature," NASA, June 2005. <http://science.hq.nasa.gov/oceans/physical/SST.html>

[55] Stoneking, Eric and Hosler, Jeff. "Path Planning Algorithms for the Adaptive Sensor Fleet," NASA AAS 05-024. 2005.

[56] Tang, Hao and Miller-Hooks, Elise. "A TABU search heuristic for the team orienteering problem," *Computers & Operations Research 32*, 1379-1407, 2005.

[57] Tang, Hao and Miller-Hooks, Elise. "Algorithms for a stochastic selective travelling salesperson problem." *The Journal of the Operational Research Society*, Vol. 56, pp. 439-452, 2005.

[58] Tang, Lixin and Wang, Xianpeng. "Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem." *International Journal of Advanced Manufacturing Technology*. Vol. 29, pp. 1246-1258, 2006.

[59] Talabac, Stephen J. "Harmful Algal Blooms (HABs): Candidate Science Scenario for a Reconfigurable Sensor Web Observing System," NASA Goddard Space Flight Center. Sep. 2003.

[60] "Telesupervised Adaptive Ocean Sensor Fleet," Central Michigan University, Sep. 2006. <http://www.cs.cmu.edu/afs/cs/user/gwp/www/TAOSF/>

[61] *The Traveling Salesman Problem and Its Variations*. Edited by Gregory Gutin and Abraham P. Punnen. Kluwer Academic Publishers Dordrecht, 2002.

[62] Tsiligirides, T., "Heuristic methods applied to orienteering." *The Journal of the Operational Research Society*, Vol. 35, No. 9. (Sep. 1984), pp. 797-809.

[63] Tsitsiklis, John. "Special Cases of Traveling Salesman and Repairman Problems with Time Windows." *Networks*, 22:263-282, 1992.

[64] Wang, Q., Sun, X., Golden, B. L., and Jia, J., "Using Artificial Neural Networks to Solve the Orienteering Problem", *Annals of Operations Research 61*, 111-120. 1995.

[65] Wertheim, Eric. "Combat Fleets." *Combat Fleets of the World Proceedings*, July 2005. <http://www.navalinstitute.org/Proceedings/Articles05/Pro07cfleets.htm>

[66] Whitley, D. "A Genetic Algorithm Tutorial," Technical Report CS-93-103, *Department of Computer Science*, Colorado State University, Aug. 1993.

[67] Wren, A., and Holiday, A., "Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points," *Operations Research Quarterly* 23, 333-344, 1972.