The Dissertation Committee for Fang Lu
certifies that this is the approved version of the following dissertation:

Modeling and Optimization for Spatial Detection to Minimize Abandonment Rate

Committee:

David P. Morton, Supervisor

John J. Hasenbein, Supervisor

James E. Bickel

Constantine Caramanis

Erhan Kutanoglu

Modeling and Optimization for Spatial Detection to Minimize Abandonment Rate

by

Fang Lu, B.E.; M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2014

Dedicated to my husband Chao, and the little angel we are expecting.

Acknowledgments

Working towards a Ph.D. is never easy, but I consider myself lucky for having the opportunity to meet with all the talented and amazing people during these five years. And I want to take this opportunity to express my gratitude for all of them.

I will forever be thankful to my supervisors, Professors David Morton and John Hasenbein. I don't think I can finish my Ph.D. without their support and guidance during the past five years. Dave is someone you will love once you meet him. Even though he is a genius, he is always so patient no matter how stupid my questions are. John is the funniest supervisor and also one of the smartest people that I know. Dave and John are the best supervisors that anyone can ask and I am very proud to be their student. I am also very grateful to Professor Alexandra Newman for helping me to improve the solution time of my mathematical models. Also, I want to thank my committee members, Professors Erhan Kutanoglu, Eric Bickel and Constantine Caramanis, for their insightful suggestions on my research.

I thank Mr. Soofi Khalid from ConocoPhillips, for introducing this great project to me and also helping me along the way. Also, I want to say thank you to my colleague and friend Zhichao Shu. We worked on a project together for one year and he gave me lots of insights and inspirations. I also

thank all my friends for being in my life to make it so colorful. There are too many names to list here but you know I am talking about you!

My family is the safe harbour for me and I am lucky to have the best parents in the world, who give me enough freedom and support to pursue my dreams. Also, special thanks to my husband, my soul mate and my best friend Chao, thank you for always being on my side and having faith in me.

Modeling and Optimization for Spatial Detection to Minimize Abandonment Rate

Publication No. _____

Fang Lu, Ph.D.
The University of Texas at Austin, 2014

Supervisors: David P. Morton
John J. Hasenbein

Some oil and gas companies are drilling and developing fields in the Arctic Ocean, which has an environment with sea ice called ice floes. These companies must protect their platforms from ice floe collisions. One proposal is to use a system that consists of autonomous underwater vehicles (AUVs) and docking stations. The AUVs measure the under-water topography of the ice floes, while the docking stations launch the AUVs and recharge their batteries. Given resource constraints, we optimize quantities and locations for the docking stations and the AUVs, as well as the AUV scheduling policies, in order to provide the maximum protection level for the platform. We first use an queueing approach to model the problem as a queueing system with abandonments, with the objective to minimize the abandonment probability. Both $M/M/k+M$ and $M/G/k+G$ queueing approximations are applied and we also develop a detailed simulation model based on the queueing

approximation. In a complementary approach, we model the system using a multi-stage stochastic facility location problem in order to optimize the docking station locations, the AUV allocations, and the scheduling policies of the AUVs. A two-stage stochastic facility location problem and several efficient online scheduling heuristics are developed to provide lower bounds and upper bounds for the multi-stage model, and also to solve large-scale instances of the optimization model. Even though the model is motivated by an oil industry project, most of the modeling and optimization methods apply more broadly to any radial detection problems with queueing dynamics.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xi**

List of Figures **xii**

Chapter 1. Introduction **1**
1.1 Background 1
1.2 Stochastic Models 5
1.3 Optimization Models 7

Chapter 2. Queueing Approximation and Simulation **10**
2.1 Queueing System with Abandonments 10
2.2 Basic Queueing Approach: $M/M/k+M$ Queue 12
2.2.1 Erlang-A Formula 13
2.2.2 Water-Filling Algorithm 15
2.3 $M/G/k+G$ Queueing Approximation 19
2.3.1 Arrival Process Simulation 20
2.3.2 General Distribution Analysis 24
2.3.3 Going Beyond the Erlang-A Formula 31
2.4 System Simulation and Computational Analysis 34

Chapter 3. Stochastic Facility Location Problem **45**
3.1 Facility Location Problem Introduction 45
3.2 Multi-Stage Stochastic Facility Location Problem 46
3.3 Simplified Docking Station Location Problem 53
3.4 Two-Stage Stochastic Facility Location Problem 57

3.5 Heuristics and Scheduling Policies 64
3.5.1 Minimize the Total Number of Tardy Jobs 66
3.5.2 Minimizing the Weighted Total Number of Tardy Jobs . 74

Chapter 4. Computational Analysis **79**
4.1 Input Parameters 79
4.2 Strategies to Reduce MIP Solution Time 86
4.3 Optimality Gap 97
4.4 Performance of the Scheduling Heuristics 101
4.5 Facility Location Problem without Queueing Dynamics 104

Chapter 5. Conclusions and Future Work **112**
5.1 Conclusions 112
5.2 Future Work 114

Bibliography **117**

List of Tables

2.1 Simulation and queueing approximations with high traffic intensity. 39

2.2 Simulation and queueing approximations with medium traffic intensity. 40

2.3 Simulation and queueing approximations with low traffic intensity. 41

2.4 Simulation with different queueing policies during high traffic intensity. 42

2.5 Simulation with different queueing policies during medium traffic intensity. 43

2.6 Simulation with different queueing policies during low traffic intensity. 44

4.1 Strategies to improve MIP solution time for model (3.2). . . . 92

4.1 Table 4.1 cont. 93

4.2 Statistics of the optimality gaps. Here $T = 250$ hours, $|\Omega| = 2$, and $u_q = 0.5$ hours for model (3.2). We change the AUV number h from 2 to 6 to achieve different traffic intensity levels, and the docking station number $m = 2$ or $m = 3$ in the multi-station case. 99

4.3 Scheduling heuristics recommendation. 105

List of Figures

2.1 We use parameters θ and α to simulate an arrival's velocity. The origin of coordinates is the platform and the alert zone radius is R. θ generates the intersection point on the circumference of the alert zone, and α determines the angle of the arrival trajectory. 21

2.2 Examples of three arrivals' stochastic trajectories, which are generated by the same distribution. 23

2.3 The ice floe enters the arrival circle on point P_2, and waits for t_{max} time periods until it moves to point B, where t_{max} is the abandonment tolerance time. The docking station sends out an AUV, which travels distance D_{out} to meet the ice floe at point C. After the scanning process, the ice floe moves to point D, and the AUV travels distance D_{in} to go back to the docking station and recharge its battery. The ice floe arrives at the alert zone when the service process finishes. 27

2.4 The ice floe enters the arrival circle on point P_2, and waits for t time periods until it moves to point B. The docking station sends out an AUV, which travels distance D_{out} to meet the ice floe at point C. After the scanning process, the ice floe moves to point D, and the AUV travels distance D_{in} to go back to the docking station and recharge its battery. The ice floe arrives at point F when the service process finishes. 29

2.5 Ice floe abandonment tolerance time histogram. 30

2.6 Ice floe service time histogram. 31

2.7 Flow chart for the Arena simulation model. 35

3.1 The figure depicts the platform, potential locations for docking stations, the alert zone, and the arrival circle. An AUV can be dispatched from a docking station to scan an ice floe only after the floe reaches the arrival circle. And, the AUV should complete scanning and uploading the data at its docking station prior to the floe reaching the alert zone. 47

3.2 An example of a scenario tree in a multi-stage model. Design decisions are made in stage 0 to locate docking stations and assign AUVs to those stations. Operation decisions are made in the subsequent stages to dispatch AUVs to scan arriving ice floes. The figure depicts a high (H) and low (L) number of ice floes arriving in each stage. 49

3.3 Relationship between the expected one-way travel time and the ratio of radii, η. We see from the figures that as the number of stations grows the optimal radius of the docking-station circle grows quickly toward the radius of the alert zone. 55

3.4 The left-hand figure shows two optimally-located docking stations. The inner dashed circle denotes the alert zone and the outer boundary indicates the point at which an AUV meets an arriving flow. The values on the axes denote nautical miles relative to the platform in the center. 56

3.5 An AUV travels distance D_1 (D_3) from the docking station to meet the ice floe and travels distance D_2 (D_4) to return to its docking station if the service process starts at time t (τ). . . . 63

4.1 Potential docking station locations on two concentric circles. . 80

4.2 Potential docking station locations on one circle. 82

4.3 Normalized optimal value from two-stage model (3.2) with fixed design decisions ($y_3 = 1$, $x_{3,0} = 4$; $y_i = 0$, $x_{i,0} = 0$, $\forall i \neq 3$, see Figure 4.2). The y-axis corresponds to $\hat{z}_q^* / \sum_{\omega \in \Omega} p^\omega |J^\omega|$, the normalized optimal value of model (3.2). The x-axis corresponds to u_q, the basic time unit. 83

4.4 Results from Heuristic 2 with different time horizon T. 84

4.5 Efficient frontier depicting the tradeoff between the abandonment rate and the total number of AUVs. 86

4.6 Optimality gaps with one docking station. 98

4.7 Relative optimality gaps for one docking station. 100

4.8 Relative optimality gaps for multiple docking stations. 100

4.9 Computational performance for Heuristics 1, 2, and 3. The design decisions are fixed: $y_3 = 1$, $y_i = 0$, $x_{i,0} = 0$, $\forall i \neq 3$, and we change $x_{3,0}$ from 1 to 6 to change the traffic intensity; see Figure 4.2 for the potential docking station locations 103

4.10 Computational performance for Heuristics 4, 5, 6, and 7. The design decisions are fixed: $y_3 = 1$, $y_i = 0$, $x_{i,0} = 0$, $\forall i \neq 3$, and we change $x_{3,0}$ from 1 to 6 to change the traffic intensity; see Figure 4.2 for the potential docking station locations 104

4.11 Abandonment rates from Heuristic 2 using different design decisions from model (3.2) and model (4.5). Here we generate 10 i.i.d. data sets and change the traffic intensity by changing the AUV quantity h. 108

4.12 Seven potential docking station locations. 109

4.13 The green pentagon shows the design decision from model (3.2) while the black circle shows the design decision from model (4.5) with $m = 1$ docking station and $h = 1$ AUV for four i.i.d. data sets. 110

4.14 Relative optimality gap performance for model (3.2) and model (4.5). Here we generate 10 i.i.d. data sets and change the traffic intensity from high to low by changing the AUV quantity, h. The relative optimality gap increases as the traffic intensity decreases. 111

Chapter 1

Introduction

1.1 Background

Since oil was first discovered near Alaska's North Slope in 1968, there has been significant interest in exploring for oil and gas in the Arctic Ocean. The United States Geological Survey (USGS) [32] estimates that 90 billion barrels of oil, 1,670 trillion cubic feet of natural gas, and 44 billion barrels of natural gas liquids are recoverable in the area north of the Arctic Circle, and the USGS also estimates that these constitute 22% of the world's recoverable resources.

With improved technology for recovering these resources, and increasing oil prices, some oil and gas companies have begun operations in the Arctic region. The Arctic region has large bodies of sea ice called ice floes, which make operating in the region challenging as the platform and other structures where operations take place must be protected. These structures are designed to resist a collision with an ice floe whose thickness and size are below certain thresholds. However, for an ice floe whose thickness or size exceeds this threshold, the collision could collapse the structure. An unexpected collapse would be life-threatening for those on the platform and also may cause a

severe oil spill and marine pollution. When a potentially threatening ice floe approaches the platform, one option is to dispatch an icebreaker to break, or divert, the floe. If the ice floe is too large for an icebreaker to handle, it would be necessary to evacuate the personnel, and to close all the drilling pipelines, which takes up to 72 hours. We call this 72-hour circle around the platform the *alert zone*.

By using satellite imaging, we can track movements of the ice floes, and measure their areas [20]. The trajectory, velocity and surface area of a floe can be updated when new satellite images are available. However, the ice floe thickness cannot be measured satisfactorily through satellite imaging at this point. A satellite can be used to measure the topography of the skyward surface of a floe, but not the under-water topography. One possibility is to use autonomous underwater vehicles (AUVs) to carry out this task. To do so requires the use of docking stations to launch and charge these AUVs, as well as upload the data they collect. The docking stations are connected to the platform, so all the uploaded information will be transmitted to the platform immediately.

When possible, it is desirable to locate the platform in relatively shallow water, but that location may be near deeper ocean depths. The docking station installation cost grows quickly as the water depth increases. Because the ocean can become much deeper outside the alert zone, it is not economical to place the docking stations there. So we need to locate the docking stations inside the alert zone. Ideally, we should measure the thickness of every arriving ice

floe and upload it to the docking station before the floe enters the alert zone. In reality, we have a budget constraint on the quantities of the AUVs and the docking stations, so the goal here is to develop mathematical models that can provide the maximum protection level for the drilling platform under the given limitations.

In our radial detection model, an AUV travels from its docking station to scan an ice floe. After that, the AUV returns to its original docking station, uploads information on the floe, and recharges its battery. An AUV cannot be dispatched to a floe if the total service time exceeds its battery life. When an ice floe is ready to be served by an AUV and all AUVs at the stations eligible to serve it are busy, the floe must "wait" for an available AUV. Of course, the floe continues to move and if the delay in service is too long, we effectively abandon the ice floe, at least with respect to scanning by an AUV, because the floe would reach the alert zone before the service process could be completed. We call this an abandonment and when such an event occurs, backup plans are used to handle the floe, e.g., sending an icebreaker to break up the floe, or to redirect, the ice floe.

In this dissertation, we present several queueing and simulation models to approximate the ice floe measurement dynamics, which we have just sketched, and also to evaluate the system performance measures. We also develop optimization models to minimize the probability of an ice floe abandonment or equivalently, the expected number of abandonments for a given set of ice floe arrival scenarios.

The ice floe arrival process is uncertain and the arrival rate varies by season. We consider two types of decisions in an attempt to minimize the ice floe abandonment rate, subject to resource limits. The *system design decisions* involve the number and location of the docking stations along with the allocations of the AUVs to each station. The *system operation decisions* include the assignment and scheduling policy by which the AUVs serve the ice floes. Specifically, the docking stations and AUVs must be located prior to observing the arrival process of the ice floes. Then, we alternately observe the ice floe arrival process for a given set of time periods and dispatch AUVs to serve those ice floes in a multi-stage setting.

Several related detection models can be found in the operations research literature. For example, Atkinson and Wein [19] study a model to detect nuclear terrorist attack of a city by locating radiation sensors, and they analyze the system via a spatial queueing model. Molyboha and Zabarankin [26] develop a stochastic optimization model to maximize the diver detection rate within a hydrophone network. Szechtman et al. [33] build several models to place a moving sensor for border surveillance with different system characteristics. We apply both queueing approximations and stochastic optimization to solve the problem, and part of the work in this dissertation can be found in different forms in papers, see [17], [8] and [7].

Even though this problem is motivated by an oil industry project, most of the modeling and optimization methodologies we use here can be applied more broadly to any radial detection models with queueing dynamics.

For example, we could envision applying the same type of nuclear detection problem introduced by Atkinson and Wein [19], where the city is similar to the drilling platform, which is the object to protect. Semi-tractor trailers driving on the roads constitute the arrivals in this case (with some subset of these trucks containing a nuclear device), and the system has similar wait-time dependent queues as our ice floe detection model. The objective is again to optimize the locations of radiation sensors in order to provide maximum security for the city, or equivalently, to minimize the abandonment rate, where an abandonment in this case can be defined as a terrorist that the system fails to detect and intercept.

1.2 Stochastic Models

We use a spatial Poisson process to approximate the arrival process of the ice floes. For simplicity, we start with an $M/M/k+M$ queueing system to characterize the abandonment feature of the detection model, where the ice floes are modeled as customers arriving to the system according to a spatial Poisson process and the AUVs are servers with an exponential service time distribution. An abandonment in this queueing system refers to the event that a customer's waiting time exceeds its tolerance, so it leaves the system without having been served. In an $M/M/k+M$ queueing system, we assume the customer's abandonment time also has an exponential distribution.

Given the locations of the docking stations, the information of the ice floes, and the allocation of AUVs at each docking station, we further

approximate the ice floe measurement process by an $M/G/k+G$ queueing system. Here the AUVs have a general service time distribution, which depends on the mechanics of scanning the ice floes, uploading data, and recharging batteries. In this $M/G/k+G$ queueing system, an ice floe's abandonment time also has a general distribution.

By employing the $M/G/k+G$ queue to approximate the spatial detection system in which AUVs serve ice floes, the probability distributions for the service and abandonment processes need to be estimated. We assume independence of the inter-arrival times, and that service times do not depend on the state of the system in both queueing approximations. That said, one key aspect of the ice floe system that an $M/M/k+M$ or an $M/G/k+G$ queueing approximation neglects is that the service time for an ice floe depends on how long the floe has waited in the queue. Because the ice floes are in motion at all times, so the length of the service process changes over time.

We can assess the extent to which, and under what conditions, this compromises the quality of an $M/G/k+G$ approximation by employing a more detailed simulation model. Further advantages of the simulation model we describe next are that it also allows us to explore a richer class of queueing policies for dispatching AUVs to serve floes, and it allows us to analyze the sensitivity of our results to changes in the underlying system parameters. We also compare the performance measures obtained from both queueing systems to the simulation results to test the quality of the approximations.

1.3 Optimization Models

The stochastic models provide us accurate estimations of the system performance measures and the simulation model can capture high fidelity system dynamics. Several stochastic optimization models are further built to answer the following questions with respect to providing the maximum protection level for the drilling platform with a limited budget:

- How many docking stations do we need and where should we put them?
- How many AUVs should be assigned to each docking station?
- What is the optimal scheduling policy for the AUVs?

In Chapter 3, we present a multi-stage stochastic facility location problem, which reflects the timing of the design decisions, the realizations of randomness, and the operation decisions. Specifically, the docking stations must be located prior to observing the arrival process of the ice floes. However, we assume the decisions governing the dispatching of AUVs to serve ice floes can be made after observing the set of ice floes to be served under a specific arrival process scenario. The objective function that we minimize is the expected total number of abandonments under a given set of scenarios.

Generally speaking, a facility location problem consists of a set of potential facility sites, where several facilities can be opened, and a set of spatially located demands to be served. The goal is to open a set of facilities, accounting for both an operational cost and a cost of system design [28]. For

example, subject to a budget constraint limiting facility installation cost, we may seek to locate facilities to minimize a weighted sum of distances from each demand point to its nearest facility. Facility location decisions affect system operation costs over a long time horizon, and so making a good decision about system design is crucial. The environment of the decision making process can be classified into three categories, see Rosenhead et al. [18]: (1) certainty, (2) risk, and (3) uncertainty. In the first situation, all the parameters are known and deterministic. Both the risk and uncertainty situations involve randomness and uncertainty in the model parameters. We call a facility location problem stochastic when, for example, the set of customers requiring service is known only through a probability distribution when we must locate the facilities, or the probability distribution may not even be available. Further model embellishments can include a time window during which each customer can be served, and objective functions such as minimizing the expected total number of unserved customers. See Snyder [31] for a detailed review of the facility location problem under uncertainty.

In facility location problems in the literature, customer service times do not change over time. In contrast, in our multi-stage stochastic facility location problem, the customer service time depends on the waiting time, since the ice floes are moving while they are "waiting" in the queue for service. Also, we capture the queueing dynamics of the system in the multi-stage model by keeping track of the AUV inventory at each docking station.

Generally, this multi-stage stochastic facility location problem is very difficult to solve directly due to the large number of decision variables and constraints. So several approximate approaches are developed in Chapter 3, including a simplified docking station location problem, a two-stage stochastic facility location problem and several online scheduling heuristics. By using the simplified docking station location problem, we can gain useful insights on the docking station locations. The two-stage stochastic facility location problem provides lower bounds for the multi-stage model with certain assumptions on the model parameters. Furthermore, the online scheduling heuristics provide upper bounds for the multi-stage model. We discuss the computational analysis and the optimality gap performance of these optimization models in Chapter 4.

Chapter 2

Queueing Approximation and Simulation

2.1 Queueing System with Abandonments

Queueing theory is the mathematical study of queues and it has been applied widely in applications as diverse as telecommunications, computer science, operations management, manufacturing and health care [6]. We focus on queueing systems with abandonments in this dissertation, and such a queueing system includes several related processes, among which there are three basic ones: (1) the *arrival process*, which describes the characteristics of the incoming customers that arrive to the system; (2) the *service process*, which consists of the scheduling policy along with the stochastic process governing the service times; (3) the *abandonment process*, which represents the impatience of the customers while they are waiting in the queue.

We can use the standard notation $A/B/k+C$ to describe a queueing system with abandonments, where A characterizes the arrival process, B characterizes the service process, k is the total number of servers, and C characterizes the abandonment process. Specifically, M means the inter-arrival, service, or the abandonment times have an exponential distribution and G indicates that these times have a general distribution. For

example, $M/G/1+M$ refers to a queueing system that has the following properties: (1) the inter-arrival times of the customers have an exponential distribution, which is equivalent to saying that the arrival process is a Poisson process; (2) the service times are generally distributed; (3) there is only one server in the system; and, (4) the abandonment times (or, patience times) are exponentially distributed.

Queueing systems with abandonments arise in multiple applications recently, such as the proper staffing levels in hospital emergency rooms in order to cope with various patient arrival rates [16], perishable goods inventory management to determine suitable queueing policies [27] and the call center staffing levels to minimize the customer abandonment rate [3]. For example, when a customer calls an inbound call center, if there is no available agent immediately, this customer has to be placed on hold for the next available agent. Some customers will end the call before an agent is available because the waiting time exceeds their tolerance. These customers are considered as the abandonments in this queueing system since they leave the system without being served. For customers, a high abandonment rate indicates high customer dissatisfaction. As such, a high abandonment rate may also lead to large economic losses. Also, the customer arrival rate may vary during different time slots. So the call centers need to determine their staffing levels according to different customer arrival rates in order to minimizing the abandonment rate. For given staffing level, they can also employ queueing theory to study different queueing policies by assigning priorities to different

types of customers. There are many studies on large scale call center queueing models with customer abandonments, in order to approximate the system performance and to improve the service quality, or equivalently, to decrease the abandonment probability of the system. See Reed [30] for a detailed review on queueing models for large scale call centers.

For the $M/M/k+M$ queue, we can do exact Markovian analysis and the Erlang-A formula provides a way to calculate the system performance measures directly [25]. But the difficulty of analysis increases when the service time distribution or the abandonment time distribution is not exponential. Lots of research work has been done in recent years in order to develop efficient approximation methods for queueing systems with abandonments, especially systems with a large number of servers [5]. In the following sections, we start with the $M/M/k+M$ queueing approximation, and then analyze both the service time and abandonment time distributions for our spatial detection model. The $M/G/k+G$ queueing approximation is then introduced and the performance of both queueing approximation methods are assessed by comparing with the simulation results.

2.2 Basic Queueing Approach: $M/M/k+M$ Queue

In this basic queueing approach, we use the $M/M/k+M$ queue due to its tractability with the Erlang-A formula. Assume we have a fixed number of docking stations placed around the drilling platform and the ice floe arrival process is a spatial Poisson process for each docking station. In both queueing

and simulation approaches, instead of analyzing all the stations simultaneously, we focus on each individual docking station. We further assume that both the AUV service time and the ice floe abandonment time are exponentially distributed. Denote the total number of AUVs at docking station i as k_i, $\forall i \in I$. Here I denotes the set of docking stations. When the ice floe arrival rate λ_i, the AUV service rate μ_i and the ice floe abandonment rate β_i are given for docking station i, we can model each docking station as an $M/M/k_i + M$ queueing system to approximate the system performance measures.

2.2.1 Erlang-A Formula

By approximating the spatial detection system using an $M/M/k_i + M$ queue, we can characterize each docking station by using the following four parameters:

- λ_i: the ice floe arrival rate for docking station i;
- μ_i: the AUV service rate at docking station i;
- k_i: the total number of AUVs at docking station i;
- β_i: the ice floe abandonment rate at docking station i.

Define $\rho_i := \lambda_i / k_i \mu_i$ as the offered work load and r_i as the utilization for each AUV at docking station i, then $r_i \leq \rho_i$ because of the abandonments. Due to the Markovian property of the model, we can use balance equations to compute the steady state distribution of the system. We briefly introduce

the Erlang-A formula below, and readers are referred to Mandelbaum and Zeltyn [25] for more detailed information.

Define the function $C(x, y)$ as:

$$C(x,y) := \frac{xe^y}{y^x} \cdot \gamma(x,y) = 1 + \sum_{j=1}^{\infty} \frac{y^j}{\prod_{p=1}^{j}(x+p)},\ x > 0,\ y \geq 0$$

where $\gamma(x,y) := \int_0^y t^{x-1}e^{-t}dt$ is the incomplete gamma function. An $M/M/k_i/k_i$ denotes a queueing system with Poisson arrival process, exponential service time, k_i servers and no waiting space. In an $M/M/k_i/k_i$ queue, if all the servers are busy when a customer arrives, the customer is blocked to the system. Define B_{k_i} as the blocking probability in the $M/M/k_i/k_i$ queueing system, and recall the Erlang-B formula [6]:

$$B_{k_i} = \frac{\left(\frac{\lambda_i}{\mu_i}\right)^{k_i}/k_i!}{\sum_{j=0}^{k_i}\left(\frac{\lambda_i}{\mu_i}\right)^{j}/j!}.$$

Then the probability of waiting $P_i(W > 0)$, the abandonment probability $P_i(AB)$, the expected waiting time $E_i(W)$ and the expected queue length $E_i(Q)$ are given as follows:

$$P_i(W>0) = \frac{C\left(\frac{k_i\mu_i}{\beta_i}, \frac{\lambda_i}{\beta_i}\right) \cdot B_{k_i}}{1 + C\left(\frac{k_i\mu_i}{\beta_i}, \frac{\lambda_i}{\beta_i} - 1\right) \cdot B_{k_i}}, \tag{2.1}$$

$$P_i(AB|W>0) = \frac{1}{\rho_i \cdot C\left(\frac{k_i\mu_i}{\beta_i}, \frac{\lambda_i}{\beta_i}\right)} + 1 - \frac{1}{\rho_i}, \tag{2.2}$$

$$P_i(AB) = P_i(AB|W > 0) \cdot P_i(W > 0), \tag{2.3}$$

$$E_i(W) = P_i(AB)/\beta_i, \tag{2.4}$$

$$E_i(Q) = \lambda_i \cdot E_i(W). \tag{2.5}$$

2.2.2 Water-Filling Algorithm

Suppose we have h available AUVs in total due to the budget limitations, and we need to allocate them to m docking stations. Assume $h \geq m$ since at least one AUV should be placed at each docking station. In this section, we propose a water-filling algorithm to calculate the optimal AUV allocation policy among the docking stations.

A water-filling algorithm is a concept that arises in communication channel design [14]. As the name water-filling suggests, it is similar to filling a container which has multiple openings with water. The water level in the container depends on the opening with the lowest height. We can prove that the water-filling algorithm is optimal as long as the objective is to minimize the maximum value of a function that is non-increasing when the AUV quantity k_i at docking station i increases.

For example, if the objective is to minimize the maximum abandonment probability among all the docking stations, then the problem can be formulated as the nonlinear program shown as model (2.6). For example, we can denote the abandonment probability $P_i(AB)$ of docking station i as $F(k_i)$. Then according to the Erlang-A formula which is described in the previous section,

$F(k_i)$ is a non-increasing function of k_i, where k_i is the total number of AUVs at docking station i. The objective function of model (2.6) is defined as $z = \max_{i \in I} F(k_i)$, so z is also a non-increasing function of k_i, $\forall i \in I$. The same property holds for many other system performance measures, e.g., the probability of waiting $P_i(W > 0)$, the expected queue length $E_i(Q)$ and the expected waiting time $E_i(W)$.

$$\min \quad z \tag{2.6a}$$

$$\text{s.t.} \quad z \geq F(k_i), \quad \forall i \in I \tag{2.6b}$$

$$\sum_{i \in I} k_i \leq h \tag{2.6c}$$

$$k_i = 0, 1, 2..., h, \quad \forall i \in I \tag{2.6d}$$

Algorithm The Water-Filling Algorithm

Input:
Total number of AUVs: h; Total number of stations: m; $\lambda_i, \mu_i, \beta_i, \forall i \in I$.

- **Step 0**: Initialize $k_i = 1$, $\forall i \in I$, $j = 0$ and $c = h - m$;
- **Step 1**: $j = j + 1$; Set $z_j^* = \max_{i \in I} F(k_i)$; $i_j^* = \arg\max_{i \in I} F(k_i)$;
- **Step 2**: if $c = 0$, $z^* = z_j^*$, **STOP**;
  Otherwise, $k_{i_j^*} = k_{i_j^*} + 1$, $c = c - 1$, go to Step 1.

Output:
AUV allocation: k_i, $\forall i \in I$; Objective value: $z^* = \max_{i \in I} F(k_i)$; Bottleneck station index: $i^* = \arg\max_{i \in I} F(k_i)$.

Here are a few comments for this water-filling algorithm:

- There are $h - m + 1$ iterations in total. The first iteration involves calculating the function $F(k_i)$ n times using the Erlang-A formula, while the other iterations only need to calculate it once. Also, we need determine $z_j^* = \max_{i \in I} F(k_i)$ and the docking station that achieves the maximum at each iteration.

- z_j^* is non-increasing at each iteration: $z_1^* \geq z_2^* \geq ... \geq z_{h-m+1}^*$ and the optimal value $z^* = z_{h-m+1}^*$.

- At each iteration, we call the docking station with the maximum $F(k_i)$ the "bottleneck" since it determines the optimal value z_j^* at that iteration.

- If there is a tie in $F(k_i)$ in any iteration, we break it in an arbitrary way.

Now we prove the optimality of the water-filling algorithm. Suppose the solution we get from the above water-filling algorithm is $(k_1, k_2, ..., k_m)$, where k_i is the total number of AUVs to be assigned to the station i and $\sum_{i=1}^m k_i = h$. We order the docking stations such that $F(k_1) \geq F(k_2) \geq \ldots \geq F(k_m)$. So the value of the objective function is $z^* = F(k_1)$ and the first station is the bottleneck station in this case.

We prove the optimality of the water-filling algorithm by contradiction. Assume the solution given by the water-filling algorithm is not optimal and instead, the optimal solution is $(\bar{k}_1, \bar{k}_2, ..., \bar{k}_m)$ with optimal value $\bar{z}^*$. Then $\sum_{i=1}^m \bar{k}_i = h$ and $\bar{z^*} < z^*$ due to the optimality of the solution. Assume

that the order of the stations is the same as the solution obtained from the water-filling algorithm, then we do not have clear relations between $F(\bar{k}_i)$ and $F(\bar{k}_j)$ in this case for $\forall i \neq j$. The first station is no longer guaranteed to be the bottleneck station in the optimal solution, so the optimal objective value $\bar{z}^*$ may or may not be achieved by the first station and we need to discuss both scenarios.

- If the first station is the bottleneck station, then $\bar{z}^* = F(\bar{k}_1)$:
  Since $\bar{z}^* = F(\bar{k}_1) < z^* = F(k_1)$, then $\bar{k}_1 > k_1$. As h is a given constant, then $\exists j \in I$ s.t. $\bar{k}_j < k_j$. So $F(k_j) < F(\bar{k}_j) \leq F(\bar{k}_1) < F(k_1)$. Since $\bar{k}_j \geq 1$, then $k_j \geq 2$. So in the water-filling algorithm, there exist $k_j - 1$ iterations for which the jth station is the bottleneck station.
  Suppose in the rth iteration of the water-filling algorithm, the jth station is the bottleneck station and there are $\bar{k}_j$ AUVs in the station at that iteration. Then $z_r^* = F(\bar{k}_j)$. As we mentioned before, z_r^* is non-increasing in r. Thus $z^* = F(k_1) \leq z_r^* = F(\bar{k}_j)$, which contradicts the conclusion above that $F(k_j) < F(\bar{k}_j) \leq F(\bar{k}_1) < F(k_1)$.
  So, there cannot exist an alternate optimal solution in which the first station is the bottleneck.

- If the first station is not the bottleneck station, and the qth ($q \neq 1$) station is the bottleneck instead, then $\bar{z}^* = F(\bar{k}_q)$:
  Also we have $F(\bar{k}_1) < F(\bar{k}_q) = \bar{z}^* < F(k_1)$, so $\bar{k}_1 > k_1$. Since h is a constant, then $\exists j \in I$ such that $\bar{k}_j < k_j$. We have $F(k_j) < F(\bar{k}_j) \leq$

$F(\bar{k}_q) < F(k_1)$. Since $\bar{k}_j \geq 1$, then $k_j \geq 2$. So in the water-filling algorithm, there exist $k_j - 1$ iterations for which the jth station is the bottleneck station. By following the analysis in the previous scenario, we can show that there cannot exist an alternate optimal solution in this case either.

We have proved that the water-filling algorithm is optimal for this $M/M/k+M$ basic queueing approach for allocating AUVs among docking stations. As a matter of fact, the water-filling algorithm is also optimal if we change the $M/M/k+M$ to an $M/G/k+G$ queueing system. The difficulty in that case is how to estimate the service time and abandonment time distributions and how to calculate the system performance measures.

2.3 $M/G/\boldsymbol{k}+G$ Queueing Approximation

In this section, we introduce the $M/G/k+G$ queueing approximation, where both the AUV service time and the ice floe abandonment time have general distributions. Given the locations of the docking stations, the velocities of the ice floe arrivals, and the total number of AUVs at each docking station, we can approximate the ice floe measurement process at each docking station as an $M/G/k+G$ queueing system, where the ice floes are modeled as customers arriving to the system according to a spatial Poisson process and the AUVs are servers with a general service time distribution which depends on the mechanics of scanning the ice floes, uploading data, and recharging batteries.

We start with the ice floe arrival process simulation, and then analyze the general distributions for the service time and the abandonment time. The $M/G/k+G$ queue is then employed to approximate the system performance measures.

2.3.1 Arrival Process Simulation

We model the ice floe arrival process using a spatial Poisson process with given arrival rate λ. Two different types of ice floe arrival processes are simulated in this model. In the first type of arrival process, we assume a deterministic and fixed velocity vector for a given arrival (although each arrival may have a different vector), which means the trajectory of an arrival is simply a deterministic straight line, given the arrival's velocity vector. In the second type of arrival process, the velocity vector for each arrival changes according to a specified distribution at certain time increments. So each arrival has a stochastic trajectory, instead of the straight-line trajectory as in the first arrival process. We first introduce the deterministic trajectory arrival process simulation methods.

Since we only need to investigate the arrival of ice floes that will enter the alert zone, we characterize a deterministic arrival trajectory of an ice floe using three parameters, θ, α and v, as follows:

- θ generates the pair $(R\sin\theta,\ R\cos\theta)$ on the circumference of the alert zone circle, where R is the radius of the alert zone. θ determines where the arrival will intersect the perimeter of the alert zone.

- α determines the direction of the arrival's velocity, $\alpha \in [0, \pi]$. The angle α is relative to the tangent of the arrival point of the ice floe on the circumference of the alert zone, as depicted in Figure 2.1.

- v represents the speed of the arrival.

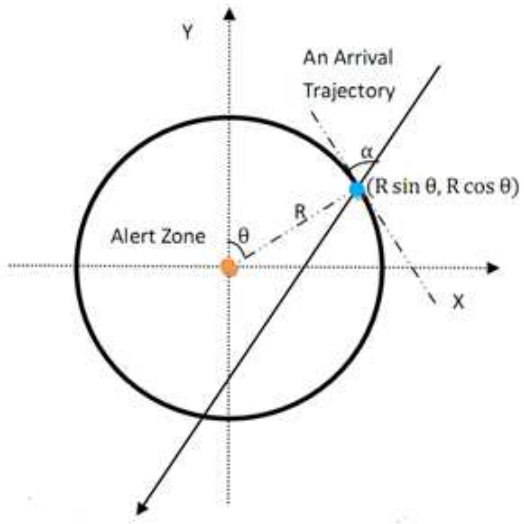

Figure 2.1: We use parameters θ and α to simulate an arrival's velocity. The origin of coordinates is the platform and the alert zone radius is R. θ generates the intersection point on the circumference of the alert zone, and α determines the angle of the arrival trajectory.

So, θ and α uniquely determine the trajectory of an arriving ice floe and v is the speed of the arrival on this trajectory. Note that we can restrict attention to α values in $(0, \pi)$ because we are only interested in arrivals that eventually enter the alert zone. The distributions of θ and α depend on the characteristics governing the movement of arrivals.

Simulating a stochastic trajectory arrival process is more complicated than simulating a deterministic one because we need to repeatedly resample both the direction and the speed of each arrival. In reality, the velocity vector changes on a continuous basis but we can only obtain satellite images every T time periods to update the locations of the ice floes and to estimate their velocities. So in order to model the movements of the ice floe arrivals, we assume each arrival's velocity vector changes according to certain distributions which we can obtain from historical data. Four parameters are needed here in order to characterize an arrival's stochastic arrival trajectory: θ, α, ι and v.

- θ and α are similar to the deterministic trajectory arrival process, except here these parameters are anchored to the arrival circle instead of the alert zone. The arrival circle is outside the alert zone and the detailed definition is given in the following paragraph. So θ and α determine the location and the direction of an ice floe when it arrives.

- ι is the current direction of the arrival, which has a probability density function f_ι. Every T time periods we generate a new movement direction of the arrival using f_ι.

- v is the speed of an arrival, but it is no longer fixed in this arrival process. Similar to the parameter ι, v is sampled from a known probability density function f_v every T time periods.

Figure 2.2 shows some examples of the ice floe trajectories generated via the scheme just described. The ice floe arrivals are initiated on the

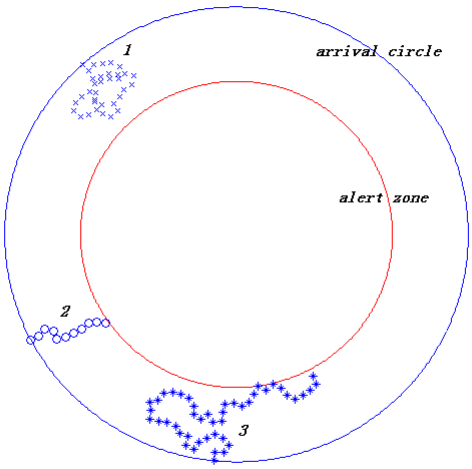

Figure 2.2: Examples of three arrivals' stochastic trajectories, which are generated by the same distribution.

arrival circle, which is defined as the circle where the ice floes are first seen or considered as arrivals to the system. Here we assume $\alpha = \pi/2$, which means all the ice floe arrivals are heading directly towards the platform when they first appear on the arrival circle. These figures exemplify the difference between the two methods of generating ice floe trajectories. In particular, even though the initial trajectory direction is directly towards the platform, some ice floes will nonetheless not enter the alert zone. As shown in Figure 2.2, we see that arrival 1 moves between the arrival circle and the alert zone, and eventually leaves the arrival circle without entering the alert zone. arrival 2 almost goes directly into the alert zone and arrival 3 finally enters the alert

zone after meandering between the two circles.

2.3.2 General Distribution Analysis

By using the arrival process simulation methods in Section 2.3.1 for the deterministic ice floe trajectories, we know an ice floe arrival enters the alert zone at the point $p_1 = (R\sin\theta, R\cos\theta)$, where R is the radius of the alert zone and we assume the center of the platform is the origin of coordinates. For given parameters θ and α, an arrival's trajectory can be characterized by the following equation:

$$y = -\tan(\theta+\alpha)x + \frac{R\cos\alpha}{\cos(\theta+\alpha)}, \quad \theta \in [0, 2\pi], \quad \alpha \in [0, \pi]. \tag{2.7}$$

We denote the radius of the arrival circle as $\hat{R}$, where we assume $\hat{R} > R$. Assuming the docking stations are uniformly placed on a circle with radius r inside the alert zone, then we have $r < R < \hat{R}$. Before entering the alert zone, an ice floe arrives at the arrival circle at point $p_2 = (\hat{R}\sin\beta, \hat{R}\cos\beta)$. Since p_2 is also a point on the ice floe's trajectory, we can plug it into (2.7) to calculate β. We get two possible values for β (assuming the domain of the arccos function is between 0 to π) as shown by (2.8) since the ice floe trajectory has two intersections with the arrival circle. If $\theta + \alpha \leq \pi$ or $2\pi < \theta + \alpha \leq 3\pi$, then $\cos\beta = \max\{\cos\beta_1, \cos\beta_2\}$; if $\pi < \theta + \alpha \leq 2\pi$, then $\cos\beta = \min\{\cos\beta_1, \cos\beta_2\}$. The parameters β_1 and β_2 can be calculated as

follows:

$$\beta_1 = \theta + \alpha + \arccos \frac{R \cos \alpha}{\hat{R}}, \tag{2.8a}$$

$$\beta_2 = \theta + \alpha - \arccos \frac{R \cos \alpha}{\hat{R}}. \tag{2.8b}$$

Here we denote the speed ratio between the AUVs and the ice floes as k ($k > 1$). The speed of the ice floe is v, as defined before, so the speed of the AUV is $k \cdot v$. We analyze the service time and abandonment time for each individual docking station. When an ice floe arrives to the system, it is assigned to the closest docking station. If an AUV is available at that station, the service process starts right away. If not, the ice floe waits in the queue until either an AUV becomes available for it, or the abandonment tolerance time is reached. There are five steps in an AUV service process:

- Step 1: An AUV goes out to meet the ice floe. We denote the distance it travels as D_{out};
- Step 2: When the AUV meets the ice floe, it starts the scanning process. The length of the scan time t_s can be influenced by several factors, such as the size of the ice floe, the density of the sea water and the speed of the ocean current. Here we assume the scan time is uniformly distributed between one and eight hours.
- Step 3: After the scanning process, the AUV obtains the thickness information of the ice floe and travels back to its original docking station.

Denote the distance it travels as D_{in}, and in general $D_{in} \neq D_{out}$ since the ice floe is moving during the scanning process;

- Step 4: When the AUV gets back to its docking station, it uploads the thickness information of the ice floe and we assume the uploading process is instantaneous;

- Step 5: The battery of the AUV needs to be recharged for four hours after each scanning sortie, so the distance that the ice floe travels is $4 \cdot v$ during the battery recharge process.

If an ice floe cannot be served before it enters the alert zone, we simply abandon the ice floe from the queue to avoid further delay. Denote the coordinates of a particular docking station that we want to analyze as $[m, n]$. The ice floe trajectory is characterized by (2.7) and we denote it as $ax + by + c = 0$ for simplicity. Then the distance between the docking station and the ice floe trajectory is $D_p = |am+bn+c|/\sqrt{a^2 + b^2}$. Also, when we draw a line that passes the docking station and is also perpendicular to the ice floe trajectory, the intersection point is $P_m = [(b^2m - ac - abn)/(a^2 + b^2), (a^2n - bc - abm)/(a^2 + b^2)]$.

Assume an AUV becomes available and starts the service process when an ice floe has waited for t_{max} time periods; then the ice floe will be on the edge of the alert zone when the service process finishes. We can write out (2.9) based on the geometry shown in Figure 2.3, and we can calculate the maximum

tolerance time t_{max} as the ratio between the distance it travels during waiting and the speed of the ice floe.

In Figure 2.3, the distance between points D and C is denoted as $D(D, C)$, and it is the distance that the ice floe travels during the scanning process, so it is a known parameter that is determined by the scanning time and the speed of the ice floe. $D(P_m, P_1)$ is the distance between P_m and the point P_1 on the alert zone, while $D(B, P_2)$ is the distance the ice floe travels while waiting in the queue. The point P_m can either be inside or outside the alert zone and our analysis holds for both cases. Then we can write out the following equations based on the geometry of the model:

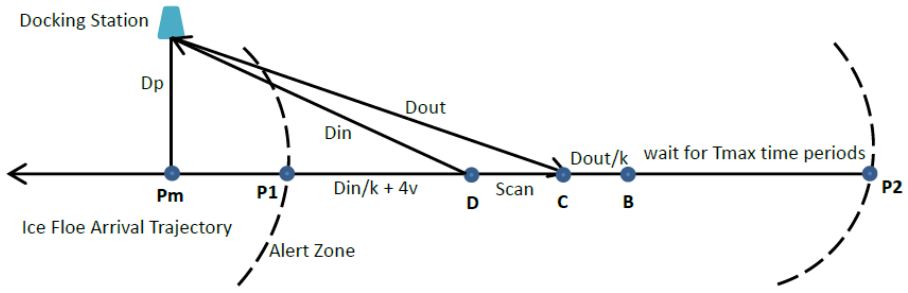

Figure 2.3: The ice floe enters the arrival circle on point P_2, and waits for t_{max} time periods until it moves to point B, where t_{max} is the abandonment tolerance time. The docking station sends out an AUV, which travels distance D_{out} to meet the ice floe at point C. After the scanning process, the ice floe moves to point D, and the AUV travels distance D_{in} to go back to the docking station and recharge its battery. The ice floe arrives at the alert zone when the service process finishes.

$$D(P_1, P_2) = \frac{D_{in}}{k} + 4v + D(D, C) + \frac{D_{out}}{k} + D(B, P_2), \tag{2.9a}$$

$$D_{in}^2 = D_p^2 + (D(P_m, P_1) + \frac{D_{in}}{k} + 4v)^2, \tag{2.9b}$$

$$D_{out}^2 = D_p^2 + (D(P_m, P_1) + \frac{D_{in}}{k} + 4v + D(D, C))^2. \tag{2.9c}$$

Solving the above equations gives us the maximum abandonment tolerance time t_{max}:

$$D_{in} = \frac{D(P_m, P_1) + 4v + \sqrt{D_p^2(k^2 - 1) + (D(P_m, P_1) + 4v)^2 k^2}}{k - 1/k}, \tag{2.10a}$$

$$D_{out} = \sqrt{D_p^2 + (D(P_m, P_1) + \frac{D_{in}}{k} + 4v + D(D, C))^2}, \tag{2.10b}$$

$$t_{max} = \frac{D(P_1, P_2) - D(D, C) - 4v - D_{in}/k - D_{out}/k}{v}. \tag{2.10c}$$

Figure 2.4 illustrates the service process when an ice floe waits for t time periods in the queue, assuming that $t \leq t_{max}$. Using similar geometric analysis for the maximum abandonment tolerance time calculation, we obtain (2.11) from Figure 2.4:

$$D(P_1, P_2) = D(P_1, F) + \frac{D_{in}}{k} + 4v + D(D, C) + \frac{D_{out}}{k} + D(P_2, B), \tag{2.11a}$$

$$D_{in}^2 = (D(P_m, P_1) + D(P_1, F) + \frac{D_{in}}{k} + 4v)^2 + D_p^2, \tag{2.11b}$$

$$D_{out}^2 = (D(P_m, P_2) - D(B, P2) - \frac{D_{out}}{k})^2 + D_p^2. \tag{2.11c}$$

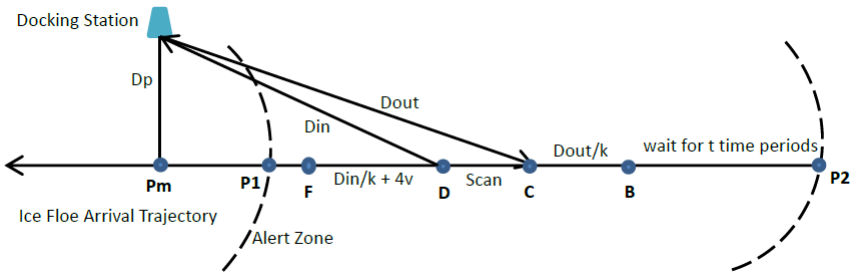

Figure 2.4: The ice floe enters the arrival circle on point P_2, and waits for t time periods until it moves to point B. The docking station sends out an AUV, which travels distance D_{out} to meet the ice floe at point C. After the scanning process, the ice floe moves to point D, and the AUV travels distance D_{in} to go back to the docking station and recharge its battery. The ice floe arrives at point F when the service process finishes.

Given the model setting and the ice floe trajectory, the unknown parameters are D_{in}, D_{out} and $D(P_1, F)$, where $D(P_1, F)$ is the distance the ice floe travels after the service process. By solving the above equations, we can obtain the following parameters and calculate the service time S:

$$D(P_m, B) = D(P_m, P_2) - D(B, P_2), \tag{2.12a}$$

$$D_{out} = \frac{-D(P_m, B) + \sqrt{D_p^2 \cdot (k^2 - 1) + D(P_m, B)^2 \cdot k^2}}{k - 1/k}, \tag{2.12b}$$

$$D_{in} = \sqrt{D_p^2 + (D(P_m, B) - D(D, C) - D_{out}/k)^2}, \tag{2.12c}$$

$$S = \frac{D_{out} + D_{in}}{k \cdot v} + t_s + 4. \tag{2.12d}$$

Assume the AUV speed is 8 knots (nautical miles per hour) and the ice floe speed $v = 0.5$ knots, then the speed ratio $k = 16$. Also, we assume $\theta \sim \mathrm{U}[0, \pi/3]$ and $\alpha \sim \mathrm{U}[0, \pi]$. The radius of the arrival circle is 50 nautical miles and the radius of the alert zone is 36 nautical miles. The location of the docking station is $[m, n] = [30\sin(\pi/6), 30\cos(\pi/6)]$. Again, the ice floe scan time is uniformly distributed between one hour to eight hours.

Using the above given parameters, we can calculate the abandonment time and the service time. Figure 2.5 shows the histogram of the ice floe abandonment time, and we can see that it is a truncated distribution with a minimum value of 10.71 hours and a maximum value of 62.83 hours. The average abandonment time is 27.91 hours and the variance is 152.34.

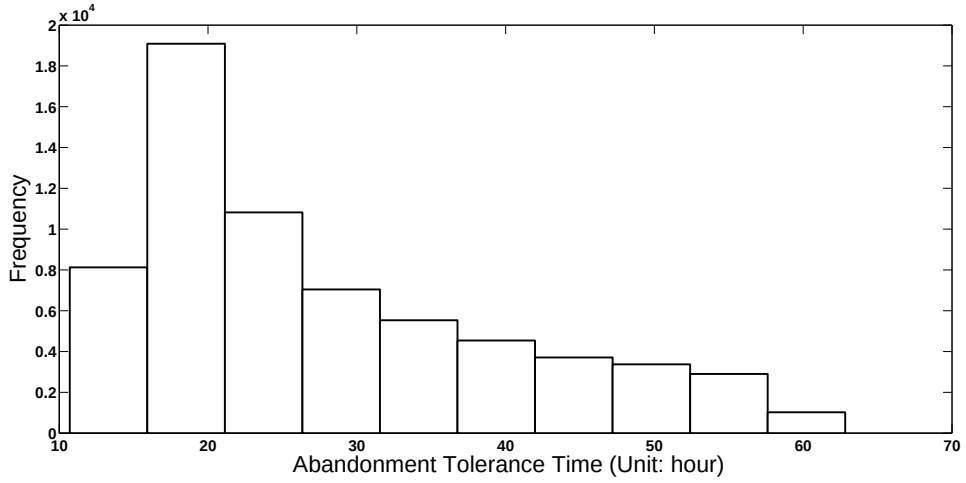

Figure 2.5: Ice floe abandonment tolerance time histogram.

Figure 2.6 shows the histogram of the ice floe service time, and here

we incorporate the wait time dependent characteristics of the service time in our analysis. For each ice floe, the service time changes while it waits in the queue, so we sample multiple service times for each ice floe and calculate the average as the approximate service time for this particular floe. The service time also has a truncated distribution. The average service time is 12.85 hours with a variance of 5.17.

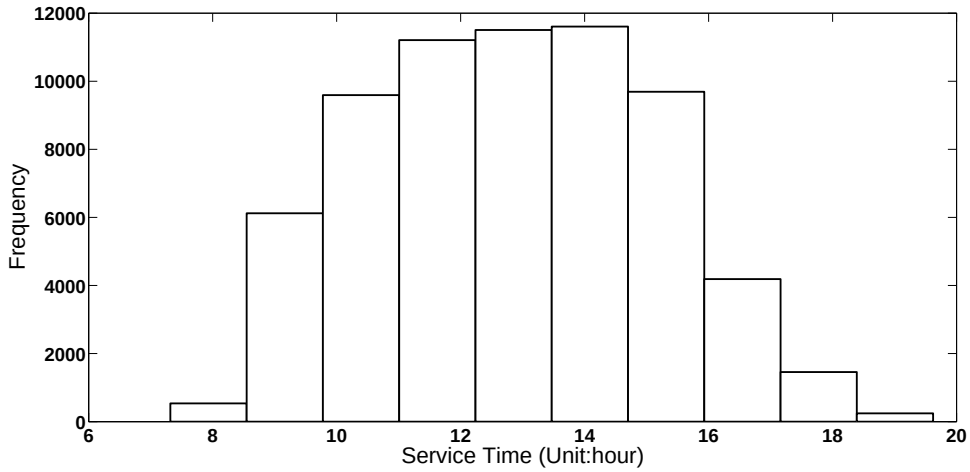

Figure 2.6: Ice floe service time histogram.

2.3.3 Going Beyond the Erlang-A Formula

The Erlang-A formula provides us useful closed form analysis for the $M/M/k+M$ queueing system. However, applying the Erlang-A formula to a $M/G/k+G$ queueing system usually yields poor approximations for the system performance measures. In general, it is very difficult to derive explicit expressions for a queueing system with general service time and abandonment

time distributions. The most common approach in the queueing literature is to develop system performance measure approximation models based on fluid or diffusion approximation [36], and most of these models are targeted to heavy traffic queueing systems with many servers [34]. Some computational research has shown that the performance measures of a $M/G/k+G$ queueing model primarily depends on the abandonment time distribution, rather than the service time distribution. So that we can sometimes approximate the $M/G/k+G$ model by the corresponding $M/M/k+G$ model [35].

Zeltyn [37] provides a detailed review on the exact analysis and many-server asymptotics of the $M/M/k+G$ queue. We adopt the approximation methodologies by Zeltyn [37] here to calculate the system performance measures for the $M/G/k+G$ queueing system. Similar to the Erlang-A model, we need the arrival rate λ_i, the service rate μ_i, and the total number of servers k_i for docking station i as the input parameters. Unlike the Erlang-A model, which only requires the mean of the abandonment time, here we need the cumulative distribution function (cdf) of the abandonment time, which is denoted as $G(\cdot)$. Denote $\bar{G}(\cdot) = 1 - G(\cdot)$ as the survival function and we define the following quantities:

$$H(x) := \int_0^x \bar{G}(u_i)du_i, \tag{2.13a}$$

$$J := \int_0^\infty \exp\{\lambda_i H(x) - k_i\mu_i x\}dx, \tag{2.13b}$$

$$J_1 := \int_0^\infty x\exp\{\lambda_i H(x) - k_i\mu_i x\}dx, \tag{2.13c}$$

$$J_H := \int_0^\infty H(x)\exp\{\lambda_i H(x) - k_i\mu_i x\}dx, \tag{2.13d}$$

$$\varepsilon := \frac{\sum_{j=0}^{k_i-1}\frac{1}{j!}\left(\frac{\lambda_i}{\mu_i}\right)^j}{\frac{1}{(k_i-1)!}\left(\frac{\lambda_i}{\mu_i}\right)^{k_i-1}}. \tag{2.13e}$$

So the probability of waiting $P_i(W > 0)$, the abandonment probability $P_i(AB)$, the expected waiting time $E_i(W)$ and the expected queue length $E_i(Q)$ for docking station i are given as follows:

$$P_i(W > 0) = \frac{\lambda_i J}{\varepsilon + \lambda_i J}\cdot\bar{G}(0), \tag{2.14a}$$

$$P_i(AB) = \frac{1 + (\lambda_i - k_i\mu_i)J}{\varepsilon + \lambda_i J}, \tag{2.14b}$$

$$E_i(W) = \frac{\lambda_i J_H}{\varepsilon + \lambda_i J}, \tag{2.14c}$$

$$E_i(Q) = \lambda_i \cdot E_i(W). \tag{2.14d}$$

We now have two queueing approximation models for the system. To assess the approximation methods, as well as to better study the system

dynamics with wait-dependent service times, we introduce a system simulation model in the following section.

2.4 System Simulation and Computational Analysis

We use the commercial software package Arena [15] to implement our simulation model. The simulation model provides accurate system performance measures for our spatial detection model, and we compare the simulation results with both the $M/M/k+M$ and the $M/G/k+G$ queueing approximation results. Also, the flexibility of the simulation model allows us to employ different queueing policies and to analyze the system sensitivity.

Figure 2.7 shows the flow chart of the simulation model. When an ice floe arrives at a docking station, an AUV, if available, is sent out to scan it. Otherwise, the ice floe waits in the queue for that particular docking station. If an AUV becomes available later for this ice floe, we update its service time since it is wait time dependent. When the waiting time exceeds the abandonment time, the ice floe is abandoned.

We define three different traffic intensity levels based on the ice floe abandonment rate. The high traffic intensity is when the ice floe abandonment rate is above 40%, the medium traffic intensity has abandonment rate between 10% and 40%, and when the ice floe abandonment rate is below 10%, we call it low traffic intensity.

Here we assume the arrival process is a spatial Poisson process with

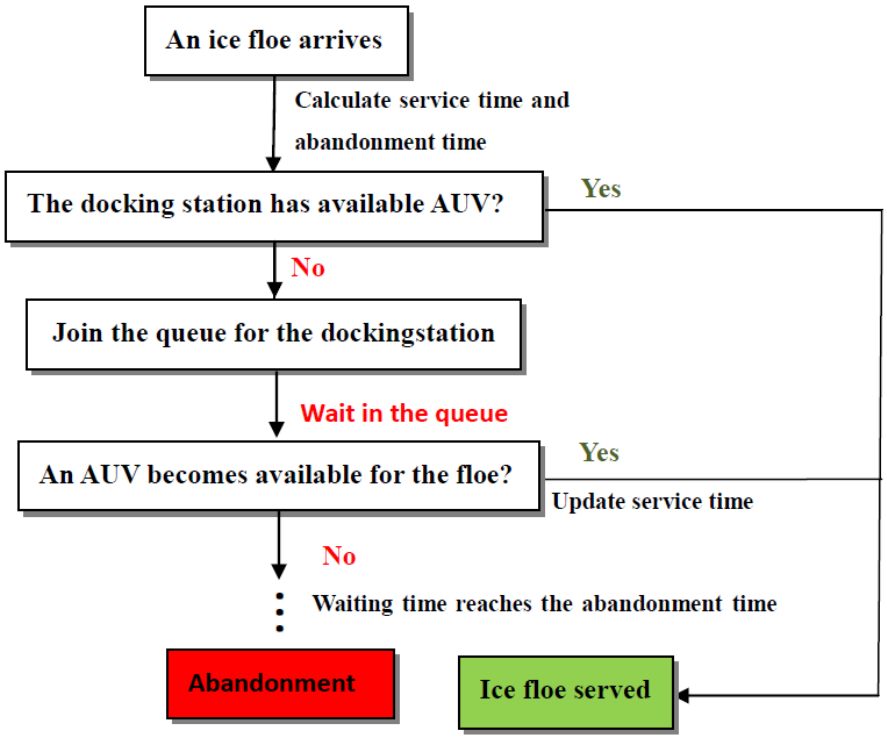

Figure 2.7: Flow chart for the Arena simulation model.

given arrival rate λ, and we assume the trajectories of all the arrivals are deterministic. For the $M/M/k + M$ queueing approximation, we apply the Erlang-A formula with the mean service time and mean abandonment tolerance time obtained from Section 2.3.2. The $M/G/k+G$ queueing model adopts the approximation methods in Section 2.3.3, and also the general distribution analysis in Section 2.3.2. We first fix the arrival rate at $\lambda = 0.5$ arrivals per hour, and change the traffic intensity level by adjusting the total number of AUVs. Later, we fix the total number of AUVs at $k = 10$, and change the arrival rate to study the model behavior at different traffic intensity

levels.

Tables 2.1, 2.2 and 2.3 show the system performance measures obtained under three different traffic intensity levels. According to Little's law, $E(Q) = \lambda \cdot E(W)$. Also, $P(AB) = \beta \cdot E(W)$ in the $M/M/k+M$ queueing model, where β is the ice floe abandonment rate.

Table 2.1 shows that the $M/G/k+G$ queueing system provides a very good approximation for all four parameters at high traffic intensity. The simulation column shows the average and 95% confidence interval for all the performance measures. For example, when the arrival rate $\lambda = 0.5$, and there are three AUVs in total, then the average probability of waiting is 0.9999, and the 95% confidence interval is $[0.9999 - 2.6468 \times 10^{-5},\ 0.9999 + 2.6468 \times 10^{-5}]$. The probability of waiting obtained from the $M/M/k+M$ queueing model is 0.9917 in this case and the relative optimality gap is 0.82%. The $M/G/k+G$ queue approximates the probability of waiting as 0.9998 and the relative optimality gap is 0.04%. The approximation results with tighter bounds are marked in bold and we can see that the $M/G/k+G$ queueing system provides better approximations in most cases, especially for the expected waiting time and the expected queue length.

During medium traffic intensity, the relative optimality gaps increase for both the $M/M/k+M$ and the $M/G/k+G$ queueing models as shown by Table 2.2. However, the $M/G/k+G$ queueing system still provides very good approximations since most of the optimality gaps are within 10%. Table 2.3 shows the model performance at low traffic intensity, and we can see the

optimality gaps for the expected waiting time and the expected queue length are still relatively small in the $M/G/k+G$ queueing approximation, even though all the optimality gaps increase when compared with the medium traffic intensity setting.

These computational results show that the relative optimality gap has different performance during different traffic intensity levels. The $M/M/k+M$ queue provides good approximations for all four parameters at high and medium traffic intensity levels, and the optimality gaps increase for both approximations when the traffic intensity decreases. The $M/M/k+M$ queue tends to overestimate the abandonment probability, and underestimate the probability of waiting, the expected waiting time and the expected queue length. In contrast, the $M/G/k+G$ approximation usually underestimate the probability of waiting, and overestimates the other three parameters.

In the above analysis, we assume both queues and the simulation model use first in first out (FIFO) as the queueing policy. If we change the queueing policy in the $M/M/k+M$ or the $M/G/k+G$ queueing system, it is generally very difficult to analyze the system performance measures. Unlike the queueing approximation, the simulation model provides us the flexibility to employ different queueing policies. Tables 2.4, 2.5, and 2.6 show the system performance with different queueing policies and traffic intensity levels.

We assume the arrival rate $\lambda = 0.5$ arrivals per hour and we change the traffic intensity by varying the total number of AUVs. We focus on four different queueing policies in this analysis:

- FIFO: first in first out,
- LIFO: last in first out,
- EDD: earliest due date first,
- SST: shortest service time first.

Table 2.4 shows the system performance measures at high traffic intensity, e.g., the abandonment probability is 0.5742 if we use LIFO as the queueing policy, and the 95% confidence interval is $[0.5742 - 0.0021, 0.5742 + 0.0021]$. For each parameter, the minimum value is marked in bold, and we can see that different queueing policies yield different model behavior. Table 2.5 shows the performance measures at medium traffic intensity and Table 2.6 shows the results at low traffic intensity. In general, the LIFO queueing policy has a low probability of waiting, expected waiting time and expected queue length when compared with the other three queueing policies. The EDD queueing policy provides the lowest abandonment probability at all traffic intensities, which is not surprising since those arrivals with the earlier due dates have higher priorities while waiting in the queue. Also, the SST queueing policy provides low expected waiting time and expected queue length.

The above analysis suggests that we should use EDD as the queueing policy for the simulation model if the objective is to minimize the abandonment rate, and LIFO is probably the best option if we aim to minimize the probability of waiting. Both LIFO and SST are good choices when we want to minimize the expected waiting time or the expected queue length.

Performance Measures	Simulation	$M/M/k+M$ Approximation	$M/G/k+G$ Approximation
Arrival Rate: $\lambda = 0.5$, No. of AUVs: $k = 3$			
$P(W > 0)$	0.9999	0.9917	**0.9995**
	$(\pm 2.6468 \times 10^{-5})$	(0.82%)	(0.04%)
$P(AB)$	0.5016	0.5346	**0.533**
	(± 0.0021)	(6.58%)	(6.26%)
$E(W)$	21.4337	14.9194	**21.809**
	(± 0.0427)	(30.39%)	(1.75%)
$E(Q)$	10.7242	7.4597	**10.9045**
	(± 0.067)	(30.44%)	(1.68%)
Arrival Rate: $\lambda = 1.5$, No. of AUVs: $k = 10$			
$P(W > 0)$	0.9999	0.9998	**0.9999**
	$(\pm 2.3598 \times 10^{-6})$	(0.01%)	(0)
$P(AB)$	0.4468	**0.481**	**0.481**
	$(\pm 8.7298e-4)$	(7.65%)	(7.65%)
$E(W)$	20.5624	13.4248	**20.7785**
	(± 0.0204)	(34.71%)	(1.05%)
$E(Q)$	30.8173	20.1372	**31.1677**
	(± 0.0767)	(34.66%)	(1.14%)

Table 2.1: Simulation and queueing approximations with high traffic intensity.

Performance Measures	Simulation	$M/M/k+M$ Approximation	$M/G/k+G$ Approximation
Arrival Rate: $\lambda = 0.5$, No. of AUVs: $k = 5$			
$P(W > 0)$	0.9934	0.8646	**0.9742**
	($\pm 6.5028 \times 10^{-4}$)	(12.97%)	(1.93%)
$P(AB)$	0.2093	0.2586	**0.2286**
	(± 0.003)	(23.55%)	(9.22%)
$E(W)$	16.0239	7.2173	**17.2255**
	(± 0.0822)	(54.96%)	(7.5%)
$E(Q)$	7.9661	3.6087	**8.6128**
	(± 0.0794)	(54.7%)	(8.12%)
Arrival Rate: $\lambda = 1$, No. of AUVs: $k = 10$			
$P(W > 0)$	0.9997	0.918	**0.9959**
	($\pm 7.9881 \times 10^{-5}$)	(8.17%)	(0.38%)
$P(AB)$	0.1973	0.2355	**0.2222**
	(± 0.0019)	(19.36%)	(12.62%)
$E(W)$	16.5833	6.5739	**17.1699**
	(± 0.0511)	(60.36%)	(3.54%)
$E(Q)$	16.5704	6.5739	**17.1699**
	(± 0.1049)	(60.33%)	(3.62%)

Table 2.2: Simulation and queueing approximations with medium traffic intensity.

Performance Measures	Simulation	$M/M/k+M$ Approximation	$M/G/k+G$ Approximation
Arrival Rate: $\lambda = 0.5$, No. of AUVs: $k = 7$			
$P(W > 0)$	0.8241	0.5304	**0.6263**
	($\pm$0.0067)	(35.64%)	(24%)
$P(AB)$	0.0254	0.0904	**0.0536**
	($\pm$0.0012)	(255.91%)	(111.02%)
$E(W)$	7.223	2.522	**8.018**
	($\pm$0.1558)	(65.08%)	(11.01%)
$E(Q)$	3.6805	1.261	**4.009**
	($\pm$0.1012)	(65.74%)	(8.93%)
Arrival Rate: $\lambda = 0.75$, No. of AUVs: $k = 10$			
$P(W > 0)$	0.9074	0.5742	**0.6986**
	($\pm$0.0048)	(36.72%)	(23.01%)
$P(AB)$	0.0286	0.0871	**0.0505**
	($\pm$0.0011)	(204.55%)	(76.57%)
$E(W)$	8.6067	2.4297	**9.3359**
	($\pm$0.1548)	(71.77%)	(8.51%)
$E(Q)$	5.8599	1.8223	**7.0019**
	($\pm$0.1555)	(68.9%)	(19.49%)

Table 2.3: Simulation and queueing approximations with low traffic intensity.

Performance Measures	Simulation (FIFO)	Simulation (LIFO)	Simulation (EDD)	Simulation (SST)
Arrival Rate: $\lambda = 0.5$, No. of AUVs: $k = 3$				
$P(W > 0)$	0.9999	**0.9992**	0.9999	0.9994
	$(\pm 2.65 \times 10^{-5})$	$(\pm 4.96 \times 10^{-5})$	$(\pm 6.50 \times 10^{-6})$	$(\pm 5.48 \times 10^{-5})$
$P(AB)$	0.5016	0.5742	0.4991	**0.4968**
	(± 0.0021)	(± 0.0021)	(± 0.0021)	(± 0.0019)
$E(W)$	21.4337	17.2536	26.656	**16.9546**
	(± 0.0427)	(± 0.053)	(± 0.0285)	(± 0.0559)
$E(Q)$	10.7242	8.6274	13.3377	**8.4797**
	(± 0.067)	(± 0.0643)	(± 0.0696)	(± 0.0641)

Table 2.4: Simulation with different queueing policies during high traffic intensity.

Performance Measures	Simulation (FIFO)	Simulation (LIFO)	Simulation (EDD)	Simulation (SST)
Arrival Rate: $\lambda = 0.5$, No. of AUVs: $k = 5$				
$P(W > 0)$	0.9934	**0.9694**	0.9983	0.9757
	$(\pm 6.5 \times 10^{-4})$	(± 0.0012)	$(\pm 2.99 \times 10^{-4})$	(± 0.0012)
$P(AB)$	0.2093	0.293	**0.1827**	0.2361
	(± 0.003)	(± 0.0031)	(± 0.0031)	(± 0.0026)
$E(W)$	16.0239	10.4619	23.2918	**10.4417**
	(± 0.0822)	(± 0.0803)	(± 0.1115)	(± 0.0752)
$E(Q)$	7.9661	**5.0761**	11.6356	5.0989
	(± 0.0794)	(± 0.066)	(± 0.1068)	(± 0.0642)

Table 2.5: Simulation with different queueing policies during medium traffic intensity.

Performance Measures	Simulation (FIFO)	Simulation (LIFO)	Simulation (EDD)	Simulation (SST)
Arrival Rate: $\lambda = 0.5$, No. of AUVs: $k = 7$				
$P(W > 0)$	0.8241	**0.7481**	0.8473	0.7633
	($\pm$0.0067)	($\pm$0.0062)	($\pm$0.0068)	($\pm$0.0064)
$P(AB)$	0.0254	0.083	**0.0063**	0.0588
	($\pm$0.0012)	($\pm$0.0022)	($\pm 6.17 \times 10^{-4}$)	($\pm$0.0016)
$E(W)$	7.223	**4.5443**	9.3555	4.599
	($\pm$0.1558)	($\pm$0.0778)	($\pm$0.2495)	($\pm$0.0841)
$E(Q)$	3.6805	**1.7021**	3.97	1.7578
	($\pm$0.1012)	($\pm$0.0502)	($\pm$0.1547)	($\pm$0.0544)

Table 2.6: Simulation with different queueing policies during low traffic intensity.

Chapter 3

Stochastic Facility Location Problem

3.1 Facility Location Problem Introduction

The queueing models introduced in Chapter 2 can approximate key system performance measures, and the discrete-event simulation model provides us the flexibility to study different queueing policies, as well as to test the quality of the queueing approximations. However, none of these models can give us optimal locations for the docking stations and the optimal AUV allocations on the docking stations. In this chapter, we formulate a stochastic facility location model, so exact optimization can be done by using these integer programming models.

In Section 3.2, we present a multi-stage stochastic facility location problem which characterizes the timing of the design decisions, the realizations of randomness, and the operation decisions. Generally, solving this integer programming model is computationally challenging due to the large problem size [9], and so we develop approximation models in the following context, including a simplified docking station location model in Section 3.3, a two-stage stochastic facility location model in Section 3.4 and several online scheduling heuristics in Section 3.5. Chapter 4 describes the computational analysis of

these approximation models, and the performance of the optimality gaps.

3.2 Multi-Stage Stochastic Facility Location Problem

In our stochastic facility location model to locate docking stations and AUVs, we assume we have a number of predetermined potential locations to place these docking stations. We further assume that we can place at most m stations; i.e., each candidate location has the same installation cost. At most n_i AUVs can be placed at location i due to a docking station's limited number of docks, and there are h AUVs in total to be positioned. Figure 3.1 shows an example of the potential locations for the docking stations, which are inside the alert zone. The arrival circle shown in Figure 3.1 is outside of the alert zone, and an AUV can only be dispatched to an ice floe once the floe crosses the arrival circle and the service time is within the AUV battery life.

A number of factors dictate the location of the arrival circle. The circle's radius should not significantly exceed the distance an AUV can travel to serve a floe based on its battery life. In addition, the future trajectory of an ice floe should not have significant variability once it reaches the arrival circle. A large arrival circle increases the flexibility for the allowable scheduling decisions, but excessive flexibility may lead to unrealistically complicated scheduling decisions that make excessive use of the specific set of arrivals. A large arrival circle also increases the complexity of the stochastic integer program, decreasing its computational tractability. A small arrival circle narrows the time window of service for each ice floe, which restricts the dispatching policies.

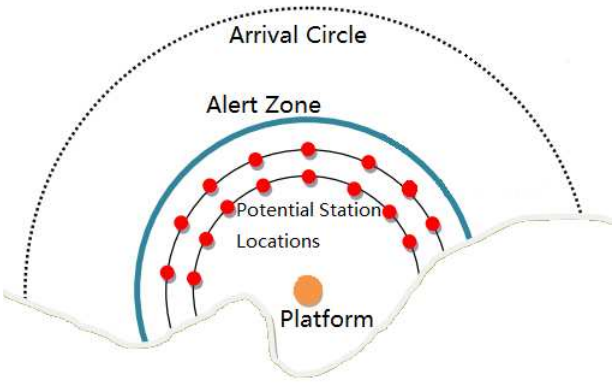

Figure 3.1: The figure depicts the platform, potential locations for docking stations, the alert zone, and the arrival circle. An AUV can be dispatched from a docking station to scan an ice floe only after the floe reaches the arrival circle. And, the AUV should complete scanning and uploading the data at its docking station prior to the floe reaching the alert zone.

Either extreme may lead to an unsatisfactory result. Within these restrictions, we have some flexibility in locating the arrival circle.

When a floe reaches the arrival circle, we calculate a time window for each floe-station pair. Any station can deploy an AUV to scan an ice floe, provided it can do so within the specified time window. An AUV can first be dispatched to scan a floe provided the service process is shorter than the AUV's battery life, and an AUV will not be dispatched if it cannot complete the service process before the ice floe enters the alert zone. An ice floe can be served by at most one AUV, and each AUV needs to return to its original docking station to upload the thickness information and recharge the battery after each scanning sortie. Satellite images can be obtained every T time

periods, which we call a stage, and we update the information regarding the ice floes at the beginning of each stage as new satellite images become available. Assume there are $K+1$ stages in total and stage 0 is the initial stage when all the design decisions are made. We now introduce the multi-stage stochastic integer programming formulation for this facility location problem.

Indices and Sets

$k = 0, 1, 2, \ldots, K$: index for the stages;

$\omega = (\omega_1, \ldots, \omega_K) \in \Omega$: set of scenarios, where ω_k is the scenario in stage k;

$\omega_{[k]} = (\omega_1, \ldots, \omega_k) \in \Omega_{[k]}$: set of scenarios up to stage k;

$t = 0, 1, \ldots, KT$: index for the time periods;

$\hat{T}_k = \{(k-1)T + 1, \ldots, kT\}$: set of time periods in stage k, $k = 1, 2, \ldots, K$;

$k(t) = \lceil t/T \rceil$: stage index for given time period t;

$i \in I$: set of potential locations for the docking stations;

$j \in J^\omega$: set of ice floes under scenario ω;

$i \in I_j^\omega$: set of locations that can serve ice floe j under scenario ω;

$j \in J_i^\omega$: set of ice floes that can be served by location i under scenario ω.

Due to the multi-stage characteristics of the model, the size of the scenario set grows exponentially as the total stage number K increases. See, for example, Heitsch and Römisch [22] for a discussion of modeling scenario trees for multi-stage stochastic programs. Figure 3.2 shows an example of the scenario tree in which there are three stages ($K = 2$), and the scenario set is $\Omega = \{HH, HL, LH, LL\}$. Stage 0 is the initial stage in which all the system design decisions are made. There is only one time period ($t = 0$) at stage

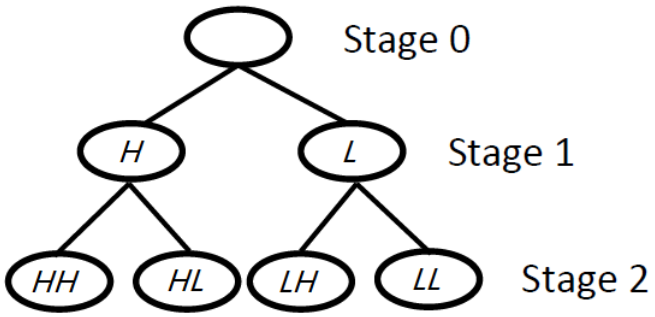

Figure 3.2: An example of a scenario tree in a multi-stage model. Design decisions are made in stage 0 to locate docking stations and assign AUVs to those stations. Operation decisions are made in the subsequent stages to dispatch AUVs to scan arriving ice floes. The figure depicts a high (H) and low (L) number of ice floes arriving in each stage.

0. The realization of scenarios starts at the beginning of stage 1, and the figure depicts two scenarios at each stage, H and L. After the realization of randomness in stage 1, the AUV scheduling decisions are made to serve the ice floes, which arrive in stage 1. After T time periods, new satellite images become available in stage 2. After observation of the new arrivals in stage 2, we again make AUV scheduling decisions for unserved floes, which have arrived so far. We define the following parameters and decision variables, and we then show the integer programming formulation in model (3.1).

Parameters

$s^{\omega}_{i,j,t}$: service time of ice floe j (hours) if served from location i at time t under scenario ω;

$[L^{\omega}_{i,j}, U^{\omega}_{i,j}]$: time window (hours) for location i to dispatch an AUV to scan floe j in scenario ω;

p^ω: probability of scenario ω;

m: maximum number of docking stations to locate;

h: maximum number of AUVs to locate;

n_i: maximum number of AUVs to locate at location i.

Decision Variables

y_i: indicates whether a station is placed at location i;

$x_{i,0}$: number of AUVs allocated to location i;

$x^\omega_{i,t}$: number of AUVs that are not dispatched at location i at time t under scenario ω;

$z^\omega_{i,j,t}$: indicates whether an AUV at location i is dispatched for floe j at time t under scenario ω;

r^ω_j: indicates whether floe j is served under scenario ω.

Multi-stage Stochastic Facility Location Problem Formulation

$$z^* = \min \sum_{\omega\in\Omega} p^\omega \sum_{j\in J^\omega} (1 - r^\omega_j) \tag{3.1a}$$

$$\text{s.t.} \sum_{i\in I^\omega_j} \sum_{t=L^\omega_{i,j}}^{U^\omega_{i,j}} z^\omega_{i,j,t} = r^\omega_j, j \in J^\omega, \omega \in \Omega \tag{3.1b}$$

$$\sum_{t=L^\omega_{i,j}}^{U^\omega_{i,j}} z^\omega_{i,j,t} \le y_i, i \in I^\omega_j, j \in J^\omega, \omega \in \Omega \tag{3.1c}$$

$$\sum_{i\in I} y_i \le m \tag{3.1d}$$

$$x_{i,0} \le n_i y_i, i \in I \tag{3.1e}$$

$$\sum_{i \in I} x_{i,0} \leq h \tag{3.1f}$$

$$x^{\omega}_{i,t} + \sum_{j \in J^{\omega}_i} z^{\omega}_{i,j,t} = x^{\omega}_{i,t-1} + \sum_{j \in J^{\omega}_i} \sum_{t'=1}^{t} z^{\omega}_{i,j,t'} \mathbb{I}(t' + s^{\omega}_{i,j,t'} = t),$$
$$i \in I, t \in \hat{T}_1 \setminus \{1\}, t \in \hat{T}_k, k = 2, \ldots, K, \omega \in \Omega \tag{3.1g}$$

$$x^{\omega}_{i,1} + \sum_{j \in J^{\omega}_i} z^{\omega}_{i,j,1} = x_{i,0} + \sum_{j \in J^{\omega}_i} z^{\omega}_{i,j,1} \mathbb{I}(s^{\omega}_{i,j,1} = 0),$$
$$i \in I, \omega \in \Omega \tag{3.1h}$$

$$x^{\omega}_{i,t} = x^{\bar{\omega}}_{i,t}, i \in I, t = 1, \ldots, KT,$$
$$\omega, \bar{\omega} \in \Omega \ (\omega_{[k(t)]} = \bar{\omega}_{[k(t)]}, \bar{\omega} \neq \omega) \tag{3.1i}$$

$$z^{\omega}_{i,j,t} = z^{\bar{\omega}}_{i,j,t}, t = L^{\omega}_{i,j}, \ldots, U^{\omega}_{i,j},$$
$$i \in I^{\omega}_j, j \in J^{\omega}, \omega, \bar{\omega} \in \Omega \ (\omega_{[k(t)]} = \bar{\omega}_{[k(t)]}, \bar{\omega} \neq \omega) \tag{3.1j}$$

$$y_i \in \{0, 1\}, i \in I \tag{3.1k}$$

$$x_{i,0} \in \{0, 1, ..., n_i\}, i \in I \tag{3.1l}$$

$$x^{\omega}_{i,t} \in \{0, 1, ..., n_i\}, i \in I, \omega \in \Omega, t \in \hat{T}_k, k = 1, 2, \ldots, K \tag{3.1m}$$

$$z^{\omega}_{i,j,t} \in \{0, 1\}, t = L^{\omega}_{i,j}, \ldots, U^{\omega}_{i,j}, i \in I^{\omega}_j, j \in J^{\omega}, \omega \in \Omega \tag{3.1n}$$

$$r^{\omega}_j \in \{0, 1\}, j \in J^{\omega}, \omega \in \Omega. \tag{3.1o}$$

We seek to minimize the expected number of ice floe abandonments via the objective function in (3.1a). Constraint (3.1b) indicates whether an ice floe has been served within its time window. Constraint (3.1c) implies we can only dispatch an AUV from location i if we have put a docking station at that location. Constraint (3.1d) limits the number of stations, (3.1e) restricts the

number of AUVs at each station, and (3.1f) bounds the total number of AUVs for all the docking stations.

Constraint (3.1g) tracks the inventory of AUVs at each time period, where the indicator function on the right-hand side of the constraint indicates whether an AUV deployed at time t' is now available after returning and recharging its battery. Constraint (3.1h) is a special case of constraint (3.1g) that differs because $x_{i,0}$ does not depend on ω, and there is only one time period in the first stage. Constraints (3.1i) and (3.1j) enforce non-anticipativity at each stage. If any two scenarios share the same history in the first k stages, then their inventory and dispatching decisions must be the same in the first k stages. These constraints are called non-anticipativity constraints in the stochastic optimization literature [4]. Finally constraints (3.1k)-(3.1o) enforce binary and integer restrictions on the decision variables.

Model (3.1) is a multi-stage stochastic facility location problem, which is computationally challenging to solve when the problem size is large. In the next section, we introduce approximations, which have lower computational complexity and can provide useful bounds for model (3.1). For reasons that become clear in the next section, we assume that the system has a basic time unit, denoted a, and all time-related parameters are multiples of a; e.g., parameter a could denote one second.

There are different methods to approximate the multi-stage stochastic optimization model. Section 3.3 simplifies the queueing dynamics and assumes radial symmetry in order to provide insight on how to locate docking stations.

We coarsen time and simplify the timing of decisions and the realization of uncertainty in a two-stage stochastic facility location problem in Section 3.4. Unlike the multi-stage model, in which we have limited information on future arrivals, the two-stage model assumes full knowledge of the arrivals after locating docking stations in the first stage. As a result, the model's scheduling decisions result in an optimistic number of ice floe abandonments. Employing coarser time units allows us to round service times and time windows in an optimistic manner. This, coupled with the optimistic assumption on knowing the future prematurely, means that the two-stage model leads to a lower bound on the optimal value, z^*, of model (3.1). When the system design decisions are fixed in the initial stage, model (3.1) reduces to a scheduling problem with the objective of minimizing the total number of abandonments. Several online scheduling heuristics are introduced in Section 3.5, and they provide upper bounds on the optimal value of model (3.1).

3.3 Simplified Docking Station Location Problem

In this section, we consider a simplified optimization model for the problem of locating docking stations. This approach gives useful insights into the placement of candidate docking station locations in a more detailed location model. In this model, we assume that docking stations must be placed on a circle inside the alert zone for reasons we discuss above regarding the cost of locating stations in deep water. And, we assume that the docking stations must be placed uniformly on this circle, which is natural if the arrival process

is radially homogeneous. In this setting we seek to optimize the radius of the docking station's circle.

Let R denote the radius of the alert zone, and r denote the radius of the docking station's circle. Then, we seek the value of $\eta = r/R$, which minimizes the expected one-way travel time of an AUV to the ice floe assuming we dispatch an AUV from the closest station at the time with the shortest travel time to the floe.

Designing the system to minimize the one-way travel time leads to an optimal design under several simplifying assumptions:

- there are ample AUVs at each station, so there are no queueing dynamics;
- an AUV can instantaneously scan an ice floe;
- every ice floe is assigned to the docking station that achieves the shortest travel time for that floe;
- the speed is the same for all the ice floes;
- ice floes arrive according to a spatial Poisson process with deterministic arrival trajectories;
- ice floes head towards the platform;
- an AUV is dispatched to an ice floe at the latest feasible time, i.e., so that data regarding the floe's thickness is uploaded just at the floe reaches the alert zone.

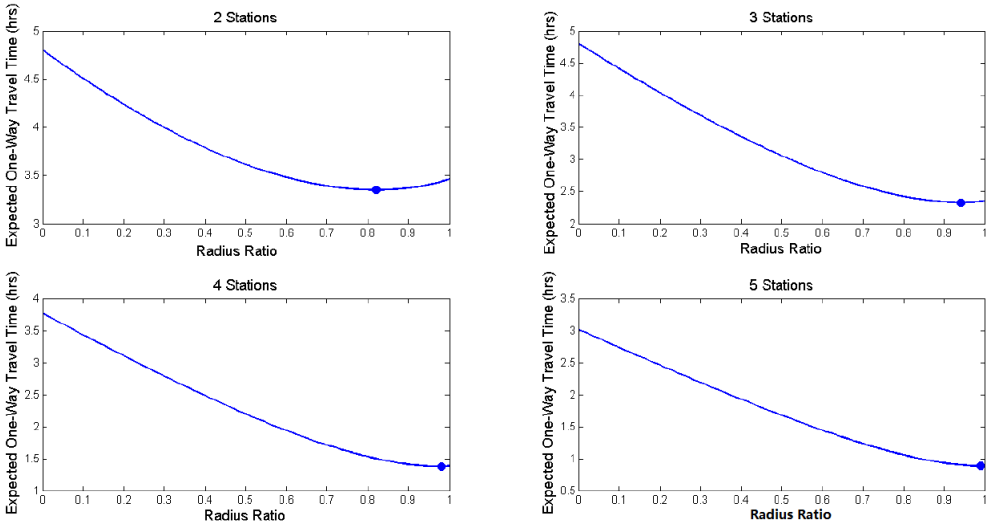

Figure 3.3: Relationship between the expected one-way travel time and the ratio of radii, η. We see from the figures that as the number of stations grows the optimal radius of the docking-station circle grows quickly toward the radius of the alert zone.

The subfigures in Figure 3.3 relate the expected one-way travel time (in hours) by an AUV to serve an ice floe and the radius ratio, η, when $m = 2, 3, 4, 5$, respectively, where m denotes the total number of docking stations. We also mark the optimal radius ratio η^*, which achieves the shortest one-way travel time in Figure 3.3. For these computations, the radius of the alert zone is assumed to be $R = 36$ nmi (nautical miles), and the speed of the ice floes is 0.5 knot (nautical mile per hour). The ratio of an AUV's speed to that of an ice floe is set to 16. Figure 3.4 depicts the locations of docking stations when we have two or three stations and we are using the optimal radius η^* obtained from Figure 3.3. The asterisks locate the docking stations and the smallest dashed circle inside is the platform we want to protect. The

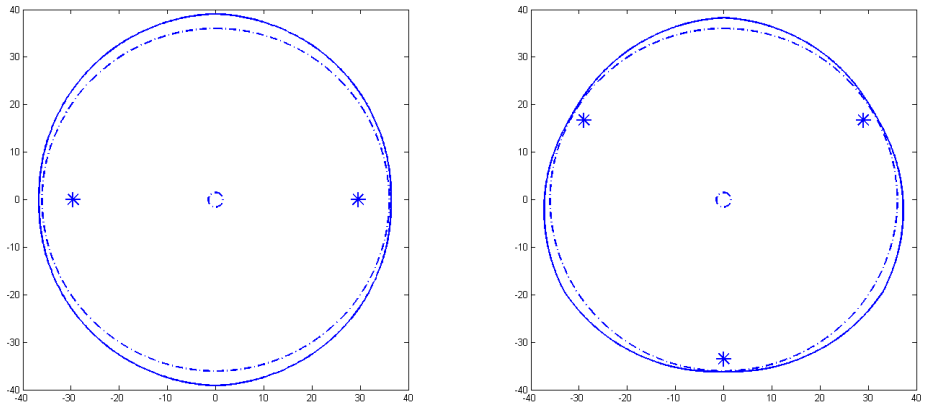

Figure 3.4: The left-hand figure shows two optimally-located docking stations. The inner dashed circle denotes the alert zone and the outer boundary indicates the point at which an AUV meets an arriving flow. The values on the axes denote nautical miles relative to the platform in the center.

large dashed circle is the alert zone, and the curve outside the alert zone is where the AUVs meet the ice floes.

Intuitively we might think the optimal strategic is to put the docking stations as close to the alert zone as possible, but this is not true according to the optimal solutions for two and three docking stations. We can prove that the optimal ratio of radii $\eta^* \to 1$ as m, which is the total number of docking stations, grows large. Also, η^* is a non-decreasing function of m.

As we can see from the above analysis, this simplified docking station location problem provides a way to calculate the docking station locations that minimize the expected travel time of the dispatched AUV. The results of this section suggests that if we locate docking stations on a concentric circle about

the platform, but within the alert zone, the radius of that circle can be close to that of the alert zone.

3.4 Two-Stage Stochastic Facility Location Problem

The two-stage stochastic facility location problem is a special case of the multi-stage model with $K = 1$. Reducing the number of stages reduces the problem size, and we further coarsen the resolution of time in the model by changing the basic time unit to be $u_q = 2^q \cdot a$, where $q \in \mathbb{Z}$. Given u_q, we round all time-related parameters in the two-stage model, and we assume that the final time period is of the form $n_q u_q + 1$ for some integer n_q for values of q of interest. Here, a is the underlying time unit used for model (3.1) as described in Section 3.2. A model with a longer basic time unit has smaller problem size but also has reduced system fidelity, since increasing the basic time unit lowers the dynamic resolution of the system. We show our two-stage stochastic facility location problem in model (3.2). Unlike model (3.1), the non-anticipativity constraints do not appear here, since the realization of randomness only happens once. Model (3.2) uses $u_q = 2^q \cdot a$ ($q \in \mathbb{Z}$) as its basic time unit, and we define the following sets and parameters for model (3.2). After stating the model we discuss bounding results based on specific means of rounding these parameters.

Sets and Parameters

$T_q = \{0, 1, u_q + 1, 2u_q + 1, \ldots, n_q u_q + 1\}$: index set for the time periods;

$L^{\omega}_{i,j,q} \in T_q$: lower bound of the time window;

$U^{\omega}_{i,j,q} \in T_q$: upper bound of the time window;

$TW^{\omega}_{i,j,q} = \{L^{\omega}_{i,j,q}, L^{\omega}_{i,j,q} + u_q, \ldots, U^{\omega}_{i,j,q} - u_q, U^{\omega}_{i,j,q}\}$: time window for location i to dispatch an AUV to scan floe j under scenario ω;

$s^{\omega}_{i,j,t,q}$: service time of ice floe j if served from location i at time t under scenario ω.

Two-Stage Stochastic Facility Location Problem Formulation

$$z^*_q = \min \sum_{\omega\in\Omega} p^{\omega} \sum_{j\in J^{\omega}} (1 - r^{\omega}_j) \tag{3.2a}$$

$$\text{s.t.} \sum_{i\in I^{\omega}_j} \sum_{t\in TW^{\omega}_{i,j,q}} z^{\omega}_{i,j,t} = r^{\omega}_j, j \in J^{\omega}, \omega \in \Omega \tag{3.2b}$$

$$\sum_{t\in TW^{\omega}_{i,j,q}} z^{\omega}_{i,j,t} \leq y_i, i \in I^{\omega}_j, j \in J^{\omega}, \omega \in \Omega \tag{3.2c}$$

$$\sum_{i\in I} y_i \leq m \tag{3.2d}$$

$$x_{i,0} \leq n_i y_i, i \in I \tag{3.2e}$$

$$\sum_{i\in I} x_{i,0} \leq h \tag{3.2f}$$

$$x^{\omega}_{i,t} + \sum_{j\in J^{\omega}_i} z^{\omega}_{i,j,t} = x^{\omega}_{i,t-u_q} + \sum_{j\in J^{\omega}_i} \sum_{\substack{t'\in T_q \\ t'\leq t}} z^{\omega}_{i,j,t'} \mathbb{I}(t' + s^{\omega}_{i,j,t',q} = t),$$
$$i \in I, t \in T_q \setminus \{0, 1\}, \omega \in \Omega \tag{3.2g}$$

$$x^{\omega}_{i,1} + \sum_{j\in J^{\omega}_i} z^{\omega}_{i,j,1} = x_{i,0} + \sum_{j\in J^{\omega}_i} z^{\omega}_{i,j,1} \mathbb{I}(s^{\omega}_{i,j,1,q} = 0),$$
$$i \in I, \omega \in \Omega \tag{3.2h}$$

$$y_i \in \{0, 1\}, i \in I \tag{3.2i}$$

$$x_{i,0} \in \{0, 1, ..., n_i\}, i \in I \tag{3.2j}$$

$$x^{\omega}_{i,t} \in \{0, 1, ..., n_i\}, i \in I, t \in T_q \setminus \{0\}, \omega \in \Omega \tag{3.2k}$$

$$z^{\omega}_{i,j,t} \in \{0,1\}, i \in I^{\omega}_{j}, j \in J^{\omega}, t \in TW^{\omega}_{i,j,q}, \omega \in \Omega \tag{3.2l}$$

$$r^{\omega}_{j} \in \{0,1\}, j \in J^{\omega}, \omega \in \Omega. \tag{3.2m}$$

For a given time horizon, model (3.2) is generally easier to solve than model (3.1). But it is still computationally challenging when the problem size grows large. Since the simplified docking station location problem in Section 3.3 provides a way to estimate the docking station locations, we temporarily restrict our attention to the operation decision variables. Denote the objective function in both models (3.1) and (3.2) as $f(X_0, Y, S)$, where X_0 and Y are the system design decisions and S stands for all the operation decisions, and let G denote the feasible region for model (3.1). Under basic time unit u_q, we let $f_q(X_0, Y, S_q)$ denote model (3.2)'s objective function, S_q denote the operation decisions and G_q denote the feasible region. Then $z^* = \min_{(X_0,Y,S)\in G} f(X_0, Y, S)$ is the optimal value for model (3.1), and similarly, $z^*_q = \min_{(X_0,Y,S_q)\in G_q} f(X_0, Y, S_q)$ is the optimal value for model (3.2). Denote the feasible region of model (3.1) with fixed (X_0, Y) by $G(X_0, Y)$, and let $G_q(X_0, Y)$ denote the analogous feasible region for model (3.2). Then we let $\hat{z}^* = \min_{S\in G(X_0,Y)} f(X_0, Y, S)$ denote the optimal value for model (3.1) with fixed design decisions (X_0, Y), and $\hat{z}^*_q = \min_{S_q\in G_q(X_0,Y)} f_q(X_0, Y, S_q)$ denotes the analog for model (3.2). Lemma 1 states that if the time related input parameters are rounded optimistically, and the service time has a relatively small change rate, increasing u_q in model (3.2) decreases $\hat{z}^*_q$.

Lemma 1. *Fix design decisions X_0 and Y in model (3.2). If $s^{\omega}_{i,j,t,q} =$*

$\left\lfloor \frac{s^{\omega}_{i,j,t}}{u_q} \right\rfloor u_q$, $L^{\omega}_{i,j,q} = \left\lfloor \frac{L^{\omega}_{i,j}-1}{u_q} \right\rfloor u_q + 1$, $U^{\omega}_{i,j,q} = \left\lceil \frac{U^{\omega}_{i,j}-1}{u_q} \right\rceil u_q + 1$, $n_q u_q + 1 = n_{q-1}u_{q-1} + 1$, *and* $|s^{\omega}_{i,j,t} - s^{\omega}_{i,j,\tau}| \leq |t - \tau|$, $\forall i \in I^{\omega}_j$, $\forall j \in J^{\omega}$, $\forall \omega \in \Omega$, $\forall t, \tau \in TW^{\omega}_{i,j,q}$, $\forall q \in \mathbb{Z}$, *then* $\hat{z}^*_{q-1} \geq \hat{z}^*_q$.

Proof. Let model M_q denote model (3.2) with basic time unit u_q and fixed design decisions X_0 and Y, $\forall q \in \mathbb{Z}$. Let S^*_{q-1} denote an optimal solution for M_{q-1}. M_q and M_{q-1} have the same final time period by hypothesis. We construct a solution S_q for M_q as follows: if $z^{\omega}_{i,j,t} = 1$ ($t \in T_{q-1} \setminus \{0\}$) in S^*_{q-1}, then set $z^{\omega}_{i,j,\tau} = 1$ ($\tau \in T_q \setminus \{0\}$) in S_q, where $\tau = \left\lfloor \frac{t-1}{u_q} \right\rfloor u_q + 1$. The decision variable r^{ω}_j is identical in both models since any served (unserved) ice floes in M_{q-1} (with solution S^*_{q-1}) remain served (unserved) in M_q (with solution S_q), so $f_q(X_0, Y, S_q) = f_{q-1}(X_0, Y, S^*_{q-1}) = \hat{z}^*_{q-1}$. It follows that $\hat{z}^*_{q-1} \geq \hat{z}^*_q$ if S_q is a feasible solution for M_q.

Given $\tau = \left\lfloor \frac{t-1}{u_q} \right\rfloor u_q + 1$ and the definitions for $L^{\omega}_{i,j,q}$ and $U^{\omega}_{i,j,q}$, if $t \in TW^{\omega}_{i,j,q-1}$ then $\tau \in TW^{\omega}_{i,j,q}$. So constraints (3.2b) and (3.2c) are satisfied in model M_q with solution S_q. As the design decisions are fixed, constraints (3.2d), (3.2e), and (3.2f) hold in model M_q. If $z^{\omega}_{i,j,t} = 1$ in solution S^*_{q-1}, then the service process for floe j starts at time t and finishes at time $\hat{t} = t + s^{\omega}_{i,j,t,q-1}$. In the corresponding schedule S_q for model M_q, $z^{\omega}_{i,j,\tau} = 1$ and the service process finishes at time $\hat{\tau} = \tau + s^{\omega}_{i,j,\tau,q}$. Since $\tau = \left\lfloor \frac{t-1}{u_q} \right\rfloor u_q + 1$, then either $t = \tau$ or $t - \tau = u_{q-1}$. If $t = \tau$ then $s^{\omega}_{i,j,t} = s^{\omega}_{i,j,\tau}$ and $s^{\omega}_{i,j,\tau,q} \leq s^{\omega}_{i,j,t,q-1}$, so we have $\hat{\tau} \leq \hat{t}$. If $t - \tau = u_{q-1}$ then $|s^{\omega}_{i,j,t} - s^{\omega}_{i,j,\tau}| \leq |t - \tau| = u_{q-1}$ by hypothesis. If $s^{\omega}_{i,j,t} \geq s^{\omega}_{i,j,\tau}$ then $s^{\omega}_{i,j,t,q-1} \geq s^{\omega}_{i,j,\tau,q}$, so $\hat{\tau} \leq \hat{t}$. If $s^{\omega}_{i,j,t} \leq s^{\omega}_{i,j,\tau}$ then

$s^{\omega}_{i,j,\tau} - s^{\omega}_{i,j,t} \leq u_{q-1}$ yields $s^{\omega}_{i,j,t,q-1} - s^{\omega}_{i,j,\tau,q} \geq -u_{q-1}$, and again we have $\hat{\tau} \leq \hat{t}$.

Consider constraint (3.2g) in M_{q-1} with solution S^*_{q-1} at time period t, where $t \in T_q \setminus \{0, 1\}$. Then we can write out the following constraints:

$$x^{\omega}_{i,t} + \sum_{j \in J^{\omega}_i} z^{\omega}_{i,j,t} = x^{\omega}_{i,t-u_{q-1}} + \sum_{j \in J^{\omega}_i} \sum_{\substack{t' \in T_{q-1} \\ t' \leq t}} z^{\omega}_{i,j,t'} \mathbb{I}(t' + s^{\omega}_{i,j,t',q-1} = t),$$

$$x^{\omega}_{i,t+u_{q-1}} + \sum_{j \in J^{\omega}_i} z^{\omega}_{i,j,t+u_{q-1}} = x^{\omega}_{i,t} + \sum_{j \in J^{\omega}_i} \sum_{\substack{t' \in T_{q-1} \\ t' \leq t+u_{q-1}}} z^{\omega}_{i,j,t'} \mathbb{I}(t' + s^{\omega}_{i,j,t',q-1} = t + u_{q-1}).$$

Summing these two equations yields:

$$x^{\omega}_{i,t+u_{q-1}} + \sum_{j \in J^{\omega}_i} (z^{\omega}_{i,j,t} + z^{\omega}_{i,j,t+u_{q-1}}) = x^{\omega}_{i,t-u_{q-1}} +$$

$$\sum_{j \in J^{\omega}_i} \Big(\sum_{\substack{t' \in T_{q-1} \\ t' \leq t}} z^{\omega}_{i,j,t'} \mathbb{I}(t' + s^{\omega}_{i,j,t',q-1} = t) + \sum_{\substack{t' \in T_{q-1} \\ t' \leq t+u_{q-1}}} z^{\omega}_{i,j,t'} \mathbb{I}(t' + s^{\omega}_{i,j,t',q-1} = t + u_{q-1}) \Big).$$

Consider time period τ in model M_q, where $\tau = t$. Then $\sum_{j \in J^{\omega}_i} (z^{\omega}_{i,j,t} + z^{\omega}_{i,j,t+u_{q-1}})$ in model M_{q-1} is equivalent to $\sum_{j \in J^{\omega}_i} z^{\omega}_{i,j,\tau}$ in model M_q. Since we have shown $\hat{\tau} \leq \hat{t}$, there exists $x^{\omega}_{i,\tau} \geq 0$ and $x^{\omega}_{i,\tau-u_q} \geq 0$ that satisfy constraint (3.2g) in model M_q. This establishes the feasibility of schedule S_q for model M_q. □

Lemma 2 shows that under the same hypotheses as lemma 1, if model (3.2) adopts the original basic time unit a from model (3.1) (or a smaller basic time unit), it achieves the largest optimal value.

Lemma 2. *Fix design decisions X_0 and Y in model (3.2). If $s^{\omega}_{i,j,t,q} = \left\lfloor \frac{s^{\omega}_{i,j,t}}{u_q} \right\rfloor u_q$, $L^{\omega}_{i,j,q} = \left\lfloor \frac{L^{\omega}_{i,j}-1}{u_q} \right\rfloor u_q + 1$, $U^{\omega}_{i,j,q} = \left\lceil \frac{U^{\omega}_{i,j}-1}{u_q} \right\rceil u_q + 1$, $n_q u_q + 1 =$*

$n_{q-1}u_{q-1}+1$, *and* $|s^{\omega}_{i,j,t}-s^{\omega}_{i,j,\tau}|\leq\ |t-\tau|$, $\forall i\in I^{\omega}_{j}$, $\forall j\in J^{\omega}$, $\forall\omega\in\Omega$, $\forall t,\tau\in TW^{\omega}_{i,j,q}$, $\forall q\in\mathbb{Z}$, *then* $\hat{z}^{*}_{q}=\hat{z}^{*}_{0}$ *if* $q\in\mathbb{Z}^{-}$, *where* $\mathbb{Z}^{-}$ *is the set of non-positive integers.*

Proof. Let model M_q denote model (3.2) with basic time unit u_q and fixed design decisions X_0 and Y, $\forall q\in\mathbb{Z}$. From Lemma 1 we have $\hat{z}^{*}_{q}\geq\hat{z}^{*}_{0}$, $\forall q\in\mathbb{Z}^{-}$. Notice that $s^{\omega}_{i,j,t,q}=s^{\omega}_{i,j,t}$, $L^{\omega}_{i,j,q}=L^{\omega}_{i,j}$, and $U^{\omega}_{i,j,q}=U^{\omega}_{i,j}$ when $q\in\mathbb{Z}^{-}$, which means the input parameters for M_q and M_0 are the same. So any scheduling solution that is feasible for M_0 is also feasible for M_q due to the finer basic time unit in M_q, which further indicates $\hat{z}^{*}_{q}\leq\hat{z}^{*}_{0}$. Combining the results we have $\hat{z}^{*}_{q}=\hat{z}^{*}_{0}$, $\forall q\in\mathbb{Z}^{-}$. □

Theorem 1. *Fix design decisions* X_0 *and* Y *in model (3.1) and model (3.2). If* $s^{\omega}_{i,j,t,q}=\left\lfloor\frac{s^{\omega}_{i,j,t}}{u_q}\right\rfloor u_q$, $L^{\omega}_{i,j,q}=\left\lfloor\frac{L^{\omega}_{i,j}-1}{u_q}\right\rfloor u_q+1$, $U^{\omega}_{i,j,q}=\left\lceil\frac{U^{\omega}_{i,j}-1}{u_q}\right\rceil u_q+1$, $n_qu_q+1=KT$, *and* $|s^{\omega}_{i,j,t}-s^{\omega}_{i,j,\tau}|\leq\ |t-\tau|$, $\forall i\in I^{\omega}_{j}$, $\forall j\in J^{\omega}$, $\forall\omega\in\Omega$, $\forall t,\tau\in TW^{\omega}_{i,j,q}$, $\forall q\in\mathbb{Z}$, *then* $\hat{z}^{*}_{q}\leq\hat{z}^{*}$, $\forall q\in\mathbb{Z}$.

Proof. With $q=0$ we have that $TW^{\omega}_{i,j,0}=\{L^{\omega}_{i,j},L^{\omega}_{i,j}+1,\ldots,U^{\omega}_{i,j}-1,U^{\omega}_{i,j}\}$. Hence, for model (3.2) with basic time unit u_0, $G(X_0,Y)\subseteq G_0(X_0,Y)$ due to the non-anticipativity constraints, and so $\hat{z}^{*}_{0}\leq\hat{z}^{*}$. Lemmas 1 and 2 show that $\hat{z}^{*}_{0}\geq\hat{z}^{*}_{q}$. So $\hat{z}^{*}_{q}\leq\hat{z}^{*}$, $\forall q\in\mathbb{Z}$. □

Theorem 2. *If* $s^{\omega}_{i,j,t,q}=\left\lfloor\frac{s^{\omega}_{i,j,t}}{u_q}\right\rfloor u_q$, $L^{\omega}_{i,j,q}=\left\lfloor\frac{L^{\omega}_{i,j}-1}{u_q}\right\rfloor u_q+1$, $U^{\omega}_{i,j,q}=\left\lceil\frac{U^{\omega}_{i,j}-1}{u_q}\right\rceil u_q+1$, $n_qu_q+1=KT$, *and* $|s^{\omega}_{i,j,t}-s^{\omega}_{i,j,\tau}|\leq\ |t-\tau|$, $\forall i\in I^{\omega}_{j}$, $\forall j\in J^{\omega}$, $\forall\omega\in\Omega$, $\forall t,\tau\in T_q\setminus\{0\}$, $\forall q\in\mathbb{Z}$, *then* $z^{*}_{q}\leq z^{*}$.

Proof. Let (X_0^*, Y^*) denote an optimal design decision for model (3.1). According to Theorem 1, $\hat{z}_q^* \leq \hat{z}^* = z^*$ if both models (3.1) and (3.2) adopt (X_0^*, Y^*) as the design decision. Also, $z_q^* \leq \hat{z}_q^*$. So $z_q^* \leq z^*$, $\forall q \in \mathbb{Z}$. □

Theorem 2 shows how to calculate a lower bound on the optimal value of model (3.1) by utilizing model (3.2), and we evaluate the computational performance in Chapter 4. In the next section, we introduce several online scheduling heuristics that provide upper bounds for model (3.1).

In the results we have presented above, we have assumed that the difference in the service times for serving floe j using an AUV from station i starting at times t and τ is bounded by the difference in those times. We close this section by showing this hypothesis holds provided the AUV speed exceeds the ice floe speed by a factor of two. Suppose the AUV speed is v_a and the speed for ice floe j is v. Then Proposition 1 shows that the change rate of the service time $s_{i,j,t}^{\omega}$ depends on the ratio between v_a and v.

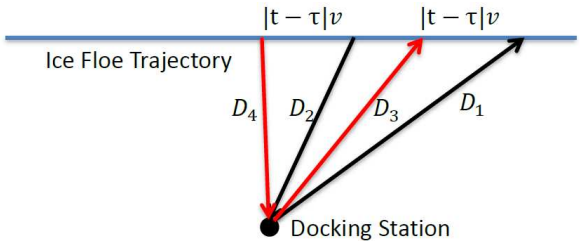

Figure 3.5: An AUV travels distance D_1 (D_3) from the docking station to meet the ice floe and travels distance D_2 (D_4) to return to its docking station if the service process starts at time t (τ).

Proposition 1. *Assume that the ice floe moves on a straight-line trajectory at speed v, and an AUV moves directly from the docking station to the floe and vice versa at speed v_a when servicing the floe. If $v_a/v \geq 2$ then $|s^{\omega}_{i,j,t} - s^{\omega}_{i,j,\tau}| \leq |t - \tau|$, $\forall i \in I^{\omega}_j$, $\forall j \in J^{\omega}$, $\forall \omega \in \Omega$, $\forall t, \tau = 1, \ldots, KT$.*

Proof. Let C denote the sum of the scan time and the AUV battery recharge time for ice floe j. Let D_1 and D_2 denote the outbound and inbound distances the AUV travels to the floe and back to the docking station if the AUV leaves the station at time t. We similarly let D_3 and D_4 depict the analogous distances if service starts at time τ, as depicted in Figure 3.5. Then, $s^{\omega}_{i,j,t} = (D_1 + D_2)/v_a + C$ and $s^{\omega}_{i,j,\tau} = (D_3 + D_4)/v_a + C$. We have $|D_1 - D_3| \leq |t - \tau|v$ and $|D_2 - D_4| \leq |t - \tau|v$ due to triangle inequality. Then $|s^{\omega}_{i,j,t} - s^{\omega}_{i,j,\tau}| = |D_1 - D_3 + D_2 - D_4|/v_a \leq |D_1 - D_3|/v_a + |D_2 - D_4|/v_a \leq 2|t - \tau|v/v_a$. So $|s^{\omega}_{i,j,t} - s^{\omega}_{i,j,\tau}| \leq |t - \tau|$ if $v_a/v \geq 2$. □

3.5 Heuristics and Scheduling Policies

In addition to the uncertainty regarding the floes requiring service, the two-stage stochastic facility location problem we present in the previous section involves scheduling dynamics associated with dispatching AUVs, presenting further computational challenges. Next, we provide an example that shows the scale of the problem size for model (3.2).

Suppose we have the following input:

- the length of the time horizon is 30 days and the basic time unit is half

an hour, so $T = 1440$;

- there are 12 potential locations to put the stations;
- there are 10 scenarios in total and around 200 ice floe arrivals per scenario;
- the time window for each ice floe is approximately 40 hours; and,
- every ice floe has a list of six stations that are eligible to serve it.

This yields a stochastic integer program for model (3.2) with 17,442,704 integer decision variables and 186,694 constraints, which will challenge current integer programming solvers. So, we seek heuristic algorithms that can approximately solve such a large scale problem efficiently.

The simplified docking station location problem gives us insights about the optimal radius of the docking station circle, which is closely related to the system design decisions in our problem. Given the locations of the docking stations, we then determine the system operation decisions, which involve scheduling the AUVs, by heuristic methods.

In the scheduling literature, a tardy job refers to a job whose service process cannot be finished by its due date, which coincides with the notion of ice floe abandonment in our model, while the objective is to minimize the total number of tardy jobs. We can also assign different priorities for different jobs, and then the objective becomes to minimize a weighted sum of the number of tardy jobs. This objective is an important performance measure in many

applications, and so it has received significant attention in the scheduling theory and a number of heuristic algorithms have been developed to minimize the weighted total number of tardy jobs under different model settings. In the next few sections, we start with the problem of minimizing the total number of tardy jobs, which corresponds to minimizing the total number of ice floe abandonments. Here, the scheduling problem perfectly captures the system dynamics of our spatial detection model. Then we assign weights for each ice floe and study the problem of minimizing the weighted total number of ice floes. Efficient scheduling heuristics are developed for both cases and they provide upper bounds for the optimal value of model (3.1).

3.5.1 Minimize the Total Number of Tardy Jobs

We start with the most basic setting of the scheduling problem here. Consider a set of jobs, which may have different due dates and processing times, that are all available at the same time, and assume there is a single server to process these jobs. In this setting, Moore's algorithm easily determines an optimal job sequence to minimize the total number of tardy jobs. See Chapter 3 of Pinedo [29] for more detailed information. However, if we assume the jobs become available at different times and there are multiple servers, the problem is more difficult to solve, and we rely on heuristics to find good solutions when the problem size grows large.

Given the docking station locations and the AUV allocations, our problem of scheduling AUVs to minimize the number of ice floe abandonments

has much in common with these scheduling problems. The ice floes correspond to jobs with different arrival times, which we denote as $L_{i,j}$ from the lower bound of the time window for ice floe j at docking station i in model (3.1). Here we suppress the scenario index ω in the notation for simplicity since it does not affect the analysis. The due date, which we denote as $U_{i,j}$, for ice floe j is the latest feasible time that an AUV in station i can be dispatched to serve it. Again, this corresponds to the upper bound of the time window for ice floe j at station i defined in model (3.1). Note that the service time $s_{i,j,t}$ depends on the ice floe waiting time and the station locations in our problem, and we only learn the information for an ice floe when it appears in the satellite images. These characteristics make the problem nonstandard and harder to solve. Several online scheduling heuristics are described in the sequel to determine the AUV dispatch policies, given the docking station locations and the number of AUVs at each station. Similar techniques have been applied in Ho and Chang [23] when the arrival times, due dates and processing times for all the jobs are static.

We assume the docking station locations, and the number of AUVs at each station, are fixed when we apply these scheduling heuristics. When new satellite images become available every T time periods, we update the ice floe profiles and run one of the following heuristics to obtain the AUV schedules. Suppose set J indexes newly arrived ice floes, which are unassigned and ready to be served. Let set J_i^c index the ice floes that have been served or are awaiting service by an AUV which has been assigned from docking station

i. We initialize $J_i^c = \emptyset$, $\forall i \in I$, when we employ the heuristics for the first time, but the heuristics subsequently take J_i^c as input. Suppose there are n_i AUVs at station i, and also define na_k, $k \in G_i$, as the next available time of AUV k at station i. Here G_i is the set of all the AUVs at docking station i. We initialize na_k as the current system time t, $\forall k \in G_i$, when we employ the heuristics for the first time, and in subsequent time periods we take na_k values as input. Also, we define st_j as the service process start time for ice floe j, which means an AUV is dispatched to serve ice floe j at time st_j. We let J^d index floes that, at least for the moment, have been declared as being abandoned. Our heuristics can attempt to reschedule AUVs to remove a floe from J^d. We use function $A(j)$ to track how many times floe j has been place in set J^d.

Scheduling Heuristic 1

- **Input:** Set I, J, G_i, current system time t, $L_{i,j}$, $U_{i,j}$, $s_{i,j,t}$, $\forall i, j, t$;
  J_i^c, na_k, $\forall k \in G_i$, $\forall i \in I$; $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \in J_i^c$;

- **Output:** J^d: set of ice floe abandonments;
  J_i^c, $\forall i \in I$: set of ice floes that have been served or are awaiting service by docking station i;
  na_k, $\forall k \in G_i$, $i \in I$: next available time for each AUV;
  $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \notin J^d$: station $\bar{\imath}_j$ dispatches AUV $\bar{k}_j$ at time st_j to serve floe j if it is not abandoned.

- **Subroutines:** Assign$(t_1, j, st_j, s_{i,j,t_1}, J_i^c)$: $\{st_j = t_1,\ t_1 = t_1 + s_{i,j,t_1},$

$J_i^c = J_i^c \cup \{j\}\}$;

Abandon$(\bar{\imath}, j, I, J^d, A(j), t, s_{i,j,t}, U_{i,j})$: $\{$if $\arg\min_{\{i \in I, i \neq \bar{\imath}\}}\{s_{i,j,t} | U_{i,j} \geq t\} = \emptyset$, $J^d = J^d \cup \{j\}$, $A(j) = A(j) + 1\}$.

- **Step 0:** Initialize $A(j) = 0$, $st_j = 0$, $\forall j \in J$; $J^d = \emptyset$, $j = 1$.

- **Step 1:** Assign floe j to station $\bar{\imath}_j \in \arg\min_{i \in I}\{s_{i,j,t} | U_{i,j} \geq t\}$. If $t > U_{i,j}$, $\forall i \in I$, then $J^d = J^d \cup \{j\}$ and go to step 3.

- **Step 2:**

  $\bar{k}_j \in \arg\min_{k \in G_{\bar{\imath}_j}}\{na_k\}$; if $na_{\bar{k}_j} \leq U_{\bar{\imath}_j, j}$, then call Assign$(na_{\bar{k}_j}, j, st_j, s_{\bar{\imath}_j, j, na_{\bar{k}_j}}, J^c_{\bar{\imath}_j})$ and go to step 3; otherwise, $A(j) = A(j) + 1$ and call Abandon$(\bar{\imath}_j, j, I, J^d, A(j), t, s_{i,j,t}, U_{i,j})$.

- **Step 3:**

  If $j = |J|$, go to step 4; otherwise, $j = j + 1$ and go to step 1.

- **Step 4:**

  Loop$(j \in J \,|\, A(j) = 1)$: $\{$assign floe j to station $\hat{\imath}_j \in \arg\min_{\{i \in I, i \neq \bar{\imath}_j\}}\{s_{i,j,t} | U_{i,j} \geq t\}$; let $\bar{\imath}_j = \hat{\imath}_j$ and $\bar{k}_j \in \arg\min_{k \in G_{\bar{\imath}_j}}\{na_k\}$. If $na_{\bar{k}_j} \leq U_{\bar{\imath}_j, j}$, then call Assign$(na_{\bar{k}_j}, j, st_j, s_{\bar{\imath}_j, j, na_{\bar{k}_j}}, J^c_{\bar{\imath}_j})$; else, $J^d = J^d \cup \{j\}\}$.

Scheduling Heuristic 2

- **Input:** Set I, J, G_i, current system time t, $L_{i,j}$, $U_{i,j}$, $s_{i,j,t}$, $\forall i, j, t$; J_i^c, na_k, $\forall k \in G_i$, $\forall i \in I$; $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \in J_i^c$;

- **Output:** J^d: set of ice floe abandonments;

  J_i^c, $\forall i \in I$: set of ice floes that have been served or are awaiting service by docking station i;

  na_k, $\forall k \in G_i$, $i \in I$: next available time for each AUV;

  $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \notin J^d$: station $\bar{\imath}_j$ dispatches AUV $\bar{k}_j$ at time st_j to serve floe j if it is not abandoned.

- **Subroutines:** Assign$(t_1, j, st_j, s_{i,j,t_1}, J_i^c)$: $\{st_j = t_1,\ t_1 = t_1 + s_{i,j,t_1},\ J_i^c = J_i^c \cup \{j\}\}$;

  Abandon$(\bar{\imath}, j, I, J^d, A(j), t, s_{i,j,t}, U_{i,j})$: $\{$if $\arg\min_{\{i \in I, i \neq \bar{\imath}\}}\{s_{i,j,t} | U_{i,j} \geq t\} = \emptyset,\ J^d = J^d \cup \{j\},\ A(j) = A(j) + 1\}$.

- **Step 0:** Initialize $A(j) = 0$, $st_j = 0$, $\forall j \in J$; $J^d = \emptyset$, $J^z = \emptyset$, $j = 1$.

- **Step 1:** Assign floe j to station $\bar{\imath}_j \in \arg\min_{i \in I}\{s_{i,j,t} | U_{i,j} \geq t\}$. If $t > U_{i,j}$, $\forall i \in I$, then $J^d = J^d \cup \{j\}$ and go to step 4.

- **Step 2:** $\bar{k}_j \in \arg\min_{k \in G_{\bar{\imath}_j}}\{na_k\}$; if $na_{\bar{k}_j} \leq U_{\bar{\imath}_j, j}$, then call Assign$(na_{\bar{k}_j}, j, st_j, s_{\bar{\imath}_j, j, na_{\bar{k}_j}}, J_{\bar{\imath}_j}^c)$ and go to step 4; else, let $\hat{z} \in \arg\max_{z \in J_{\bar{\imath}_j}^c}\{s_{\bar{\imath}_j, z, st_z} | st_z \geq t\}$, loop$(k \in G_{\bar{\imath}_j})$: $\{pna_k = na_k,\ pna_k = \min_{q \in J_{\bar{\imath}_j}^c}\{st_q | \bar{k}_q = k, st_q \geq st_{\hat{z}}\}\}$.

- **Step 3:** Loop$(q \in J_{\bar{\imath}_j}^c | st_q > st_{\hat{z}})$: $\{\bar{k}_q \in \arg\min_{k \in G_{\bar{\imath}_j}}\{pna_k\},\ pst_q = pna_{\bar{k}_q},\ pna_{\bar{k}_q} = pna_{\bar{k}_q} + s_{\bar{\imath}_j, q, pst_q}\}$, $\bar{k}_j \in \arg\min_{k \in G_{\bar{\imath}_j}}\{pna_k\}$; if $pna_{\bar{k}_j} \leq U_{\bar{\imath}_j, j}$, then call Assign$(pna_{\bar{k}_j}, j, st_j, s_{\bar{\imath}_j, j, pna_{\bar{k}_j}}, J_{\bar{\imath}_j}^c)$, loop$(k \in G_{\bar{\imath}_j}, q \in J_{\bar{\imath}_j}^c | st_q > st_{\hat{z}}, q \neq j)$: $\{na_k = pna_k,\ st_q = pst_q\}$, $J_{\bar{\imath}_j}^c = J_{\bar{\imath}_j}^c \backslash \{\hat{z}\}$,

$A(\hat{z}) = 1$, $J^z = J^z \cup \{\hat{z}\}$, and call Abandon$(\bar{\imath}_j, \hat{z}, I, J^d, A(\hat{z}), t, s_{i,\hat{z},t}, U_{i,\hat{z}})$; else, $A(j) = A(j) + 1$ and call Abandon$(\bar{\imath}_j, j, I, J^d, A(j), t, s_{i,j,t}, U_{i,j})$.

- **Step 4:** If $j = |J|$, then go to step 5; else, $j = j + 1$ and go to step 1.

- **Step 5:** Loop$(j \in J \cup J^z \,|\, A(j) = 1)$: {assign floe j to station $\hat{\imath}_j \in \arg\min_{\{i \in I, i \neq \bar{\imath}_j\}}\{s_{i,j,t} | U_{i,j} \geq t\}$; let $\bar{\imath}_j = \hat{\imath}_j$ and $\bar{k}_j \in \arg\min_{k \in G_{\bar{\imath}_j}}\{na_k\}$. If $na_{\bar{k}_j} \leq U_{\bar{\imath}_j,j}$, then call Assign$(na_{\bar{k}_j}, j, st_j, s_{\bar{\imath}_j,j,na_{\bar{k}_j}}, J^c_{\bar{\imath}_j})$; else, $J^d = J^d \cup \{j\}$}.

In both Heuristic 1 and Heuristic 2, ice floe j is assigned to the station for which it has the shortest service time, given the floe's due date using that station exceeds the current system time t. If there is no AUV that can serve floe j before its due date on that station, we reassign it in Heuristic 1. In Heuristic 2, if ice floe j cannot be served before its due date by the station with the shortest service time, we reassign the floe with the longest service time in the queue for that particular station, given that we can serve floe j by reassigning this ice floe. For the reassigned ice floe, if t is smaller than its due date then it means there may be another station that can serve the floe before its due date. So we give such an ice floe "another chance" by reassigning it to the station with the second shortest service time, given that the ice floe's due date using this station exceeds t. The function $A(j)$ restricts the total number of reassignments for floe j. The reassignment procedure is necessary here because the ice floe service time depends on its waiting time and the docking station locations. Also, each floe has a different due date using each

station. Heuristics 1 and 2 take all the AUVs in the assigned station into account and hence balance the workload of the AUVs within a station.

Scheduling Heuristic 3

- **Input:** Set I, J, G_i, current system time t, $L_{i,j}$, $U_{i,j}$, $s_{i,j,t}$, $\forall i, j, t$; J_i^c, na_k, $\forall k \in G_i$, $\forall i \in I$; $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \in J_i^c$;

- **Output:** J^d: set of ice floe abandonments;
  J_i^c, $\forall i \in I$: set of ice floes that have been served or are awaiting service by docking station i; na_k, $\forall k \in G_i$, $i \in I$: next available time for each AUV;
  $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \notin J^d$: station $\bar{\imath}_j$ dispatches AUV $\bar{k}_j$ at time st_j to serve floe j if it is not abandoned.

- **Subroutines:** Assign() and Abandon() from Algorithm 1;
  Move$(j, A(j), J^z, J_i^c)$: $\{A(j) = 1,\ J^z = J^z \cup \{j\},\ J_i^c = J_i^c \backslash \{j\}\}$.

- **Step 0:** Initialize $A(j) = 0$, $st_j = 0$, $\forall j \in J$; $J^d = \emptyset$, $J^z = \emptyset$, $j = 1$.

- **Step 1:** Assign floe j to station $\bar{\imath}_j \in \arg\min_{i \in I}\{s_{i,j,t} | U_{i,j} \geq t\}$. If $t > U_{i,j}$, $\forall i \in I$, then $J^d = J^d \cup \{j\}$ and go to step 4. Loop$(k \in G_{\bar{\imath}_j})$:$\{na_k = \max\{t, \min_{q \in J_{\bar{\imath}_j}^c}\{st_q | \bar{k}_q = k, st_q \geq t\}\}\}$, $J_{\bar{\imath}_j}^c = J_{\bar{\imath}_j}^c \cup \{j\}$, $st_j = t$, loop$(q \in J_{\bar{\imath}_j}^c | st_q \geq t)$: {order the floes according to EDD}, $J^s = \emptyset$.

- **Step 2:** Loop$(k \in G_{\bar{\imath}_j})$:{loop$(q \in J_{\bar{\imath}_j}^c \mid q \notin J^s,\ st_q \geq t)$:{If $na_k \leq U_{\bar{\imath}_j,q}$, then $\bar{k}_q = k$ and call Assign$(na_k, q, st_q, s_{\bar{\imath}_j,q,na_k}, J^s)$. If $na_k > U_{\bar{\imath}_j,q}$, then

$\hat{z} \in \arg\max_{z \in J^s}\{s_{\bar{\imath}_j,z,st_z} | \bar{k}_z = k\}$, $pna_k = st_{\hat{z}}$ and loop($r \in J^s \mid \bar{k}_r = k$, $st_r > st_{\hat{z}}$): $\{pst_r = pna_k$, $pna_k = pna_k + s_{\bar{\imath}_j,r,pst_r}\}$;
if $pna_k \leq U_{\bar{\imath}_j,q}$, then call Assign($pna_k, q, st_q, s_{\bar{\imath}_j,q,pna_k}, J^s$), $na_k = pna_k$, loop($r \in J^s \mid \bar{k}_r = k$, $st_r > st_{\hat{z}}$): $\{st_r = pst_r\}$, $J^s = J^s \backslash \{\hat{z}\}$, and if $k = |G_{\bar{\imath}_j}|$, call Move($\hat{z}, A(\hat{z}), J^z, J^c_{\bar{\imath}_j}$) and Abandon($\bar{\imath}_j, \hat{z}, I, J^d, A(\hat{z}), t, s_{i,\hat{z},t}, U_{i,\hat{z}}$); if $pna_k > U_{\bar{\imath}_j,q}$ and $k = |G_{\bar{\imath}_j}|$, then call Move($q, A(q), J^z, J^c_{\bar{\imath}_j}$) and Abandon($\bar{\imath}_j, q, I, J^d, A(q), t, s_{i,q,t}, U_{i,q}$).}}

- **Step 3:** If $j = |J|$, then go to step 4; else, $j = j + 1$ and go to step 1.

- **Step 4:** Loop($j \in J \cup J^z \mid A(j) = 1$): {assign floe j to station $\hat{\imath}_j \in \arg\min_{\{i \in I, i \neq \bar{\imath}_j\}}\{s_{i,j,t} | U_{i,j} \geq t\}$; let $\bar{\imath}_j = \hat{\imath}_j$ and $\bar{k}_j \in \arg\min_{k \in G_{\bar{\imath}_j}}\{na_k\}$. If $na_{\bar{k}_j} \leq U_{\bar{\imath}_j,j}$, then call Assign($na_{\bar{k}_j}, j, st_j, s_{\bar{\imath}_j,j,na_{\bar{k}_j}}, J^c_{\bar{\imath}_j}$); else, $J^d = J^d \cup \{j\}$}.

As in Heuristics 1 and 2, Heuristic 3 assigns each unassigned ice floe j to the station $\bar{\imath}_j$ that achieves the shortest service time, given the floe has not reached its due date for that station. Let $J^c_{\bar{\imath}_j}$ index the ice floes that are awaiting service by an AUV from docking station $\bar{\imath}_j$, and order them according to earliest due date (EDD); i.e., order them based on their U_{ij} values. We break the $n_{\bar{\imath}_j}$–AUV scheduling problem at station $\bar{\imath}_j$ into $n_{\bar{\imath}_j}$ one–AUV scheduling problems and solve them one by one, where $n_{\bar{\imath}_j}$ is the total number of AUVs at station $\bar{\imath}_j$. The one–AUV problem can be solved by Moore's algorithm, which is described in Step 2 of Heuristic 3. We employ the same reassignment

procedure as in Heuristics 1 and 2 after solving the $n_{\bar{\imath}_j}$ one–AUV scheduling problems. Heuristic 3 greedily solves a sequence of single AUV scheduling problems to minimize the number of abandonments among the remaining ice floes by using Moore's algorithm.

We can adjust the abandonment criterion by allowing multiple reassignments, which means we can put $A(j) > \hat{A}$ as the reassignment criterion in the heuristics, where $\hat{A}$ can be any finite integer threshold. We expect the abandonment rate to be non-increasing as $\hat{A}$ grows, since this essentially gives more opportunities for each ice floe to be served. However, increasing $\hat{A}$ also increases the computational complexity of the algorithm and computational experiments show little gain with respect to minimizing the total number of tardy jobs by increasing $\hat{A}$, and so we choose $\hat{A} = 1$ in all of our scheduling heuristics.

3.5.2 Minimizing the Weighted Total Number of Tardy Jobs

As we mention in the previous sections, for a given set of jobs that is all available at the beginning of the time horizon to be processed on a single server, the problem of minimizing the total number of tardy jobs is relatively easy, and we have an efficient algorithm to determine the optimal sequence of jobs. However, if we assign different weights to each job and the objective function is to minimize the weighted total number of tardy jobs, the problem becomes NP-hard. Still, there is a pseudopolynomial algorithm based on dynamic programming that can solve the problem to optimality [24].

In Chapter 2, we introduce the simulation process of the ice floe arrivals, and there are two different types of ice floe trajectories. The first approach assumes every ice floe has a deterministic trajectory, and we know its velocity after the ice floe enters the alert zone. The system restricts attention to those ice floes whose trajectories intersect the alert zone. In this case, the problem of minimizing the total number of abandonments perfectly captures the scheduling dynamics of the model. In the second approach, the ice floes have stochastic trajectories which may, or may not, intersect the alert zone. So each ice floe has a different probability of entering the alert zone, and we consider that probability as the weight w_j in this case. The objective becomes to minimize the weighted total number of ice floe abandonments. Also, the due date, $U_{i,j}$, is no longer deterministic, and it must be updated every time when we have new satellite information.

We describe four different heuristic algorithms for the problem with stochastic ice floe trajectories, and the basic structure for these algorithms is similar, except that we use different criteria to sequence the jobs. It is worth mentioning that we can also assign weights to the ice floes based on other criteria, e.g., we can give larger weights to those ice floes with longer service times, or faster speeds. Here we also assume set J contains all the unassigned ice floes between the alert zone and the arrival circle. Notice if an ice floe goes out of the arrival circle, we simply delete it out from the system. If the same ice floe reenters the arrival circle, we consider it as a new arrival.

Scheduling Heuristic 4

- **Input:** Set I, J, G_i, current system time t, $L_{i,j}$, $U_{i,j}$, $s_{i,j,t}$, $\forall i, j, t$; J_i^c, na_k, $\forall k \in G_i$, $\forall i \in I$; $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \in J_i^c$;

- **Output:** J^d: set of ice floe abandonments;
  J_i^c, $\forall i \in I$: set of ice floes that have been served or are awaiting service by docking station i; na_k, $\forall k \in G_i$, $i \in I$: next available time for each AUV;
  $\bar{\imath}_j$, $\bar{k}_j$, st_j, $\forall j \notin J^d$: station $\bar{\imath}_j$ dispatches AUV $\bar{k}_j$ at time st_j to serve floe j if it is not abandoned.

- **Subroutines:** Assign() and Abandon() from Algorithm 1;
  Move$(j, A(j), J^z, J_i^c)$: $\{A(j) = 1,\ J^z = J^z \cup \{j\},\ J_i^c = J_i^c \backslash \{j\}\}$.

- **Step 0:** Initialize $A(j) = 0$, $st_j = 0$, $\forall j \in J$; $J^d = \emptyset$, $J^z = \emptyset$, $j = 1$.

- **Step 1:** Assign floe j to station $\bar{\imath}_j \in \arg\min_{i \in I}\{s_{i,j,t} | U_{i,j} \geq t\}$. If $t > U_{i,j}$, $\forall i \in I$, then $J^d = J^d \cup \{j\}$ and go to step 4. Loop$(k \in G_{\bar{\imath}_j})$:$\{na_k = \max\{t, \min_{q \in J_{\bar{\imath}_j}^c}\{st_q | \bar{k}_q = k, st_q \geq t\}\}\}$, $J_{\bar{\imath}_j}^c = J_{\bar{\imath}_j}^c \cup \{j\}$, $st_j = t$, loop$(q \in J_{\bar{\imath}_j}^c | st_q \geq t)$: $\{w_1/(U_{\bar{\imath}_j,1} - t) \geq w_2/(U_{\bar{\imath}_j,2} - t) \geq \cdots\}$, $J^s = \emptyset$.

- **Step 2:** Loop$(k \in G_{\bar{\imath}_j})$:{loop$(q \in J_{\bar{\imath}_j}^c \mid q \notin J^s,\ st_q \geq t)$:{If $na_k \leq U_{\bar{\imath}_j,q}$, then $\bar{k}_q = k$ and call Assign$(na_k, q, st_q, s_{\bar{\imath}_j,q,na_k}, J^s)$. If $na_k > U_{\bar{\imath}_j,q}$, then $\hat{z} \in \arg\max_{z \in J^s}\{s_{\bar{\imath}_j,z,st_z} | \bar{k}_z = k\}$, $pna_k = st_{\hat{z}}$ and loop$(r \in J^s \mid \bar{k}_r = k,$ $st_r > st_{\hat{z}})$: $\{pst_r = pna_k,\ pna_k = pna_k + s_{\bar{\imath}_j,r,pst_r}\}$;
  if $pna_k \leq U_{\bar{\imath}_j,q}$, then call Assign$(pna_k, q, st_q, s_{\bar{\imath}_j,q,pna_k}, J^s)$,

$na_k = pna_k$, loop($r \in J^s \mid \bar{k}_r = k,\ st_r > st_{\hat{z}}$): $\{st_r = pst_r\}$, $J^s = J^s \backslash \{\hat{z}\}$, and if $k = |G_{\bar{\imath}_j}|$, call Move($\hat{z}, A(\hat{z}), J^z, J^c_{\bar{\imath}_j}$) and Abandon($\bar{\imath}_j, \hat{z}, I, J^d, A(\hat{z}), t, s_{i,\hat{z},t}, U_{i,\hat{z}}$); if $pna_k > U_{\bar{\imath}_j,q}$ and $k = |G_{\bar{\imath}_j}|$, then call Move($q, A(q), J^z, J^c_{\bar{\imath}_j}$) and Abandon($\bar{\imath}_j, q, I, J^d, A(q), t, s_{i,q,t}, U_{i,q}$).}}

- **Step 3:** If $j = |J|$, then go to step 4; else, $j = j + 1$ and go to step 1.

- **Step 4:** Loop($j \in J \cup J^z \mid A(j) = 1$): {assign floe j to station $\hat{\imath}_j \in \arg\min_{\{i \in I, i \neq \bar{\imath}_j\}} \{s_{i,j,t} | U_{i,j} \geq t\}$; let $\bar{\imath}_j = \hat{\imath}_j$ and $\bar{k}_j \in \arg\min_{k \in G_{\bar{\imath}_j}} \{na_k\}$. If $na_{\bar{k}_j} \leq U_{\bar{\imath}_j,j}$, then call Assign($na_{\bar{k}_j}, j, st_j, s_{\bar{\imath}_j,j,na_{\bar{k}_j}}, J^c_{\bar{\imath}_j}$); else, $J^d = J^d \cup \{j\}$}.

Heuristic 4 is very similar to Heuristic 3, except here we give priorities to those jobs with earlier due dates and larger weights, because they may be more likely to enter the alert zone. The only difference between Heuristic 4 and the following heuristics is the ordering criterion in Step 1. In particular, in each of the heuristics we give below, the rule based on ordering the floes is changed. For example, for Heuristic 5 we replace "loop($q \in J^c_{\bar{\imath}_j} | st_q \geq t$): $\{w_1/(U_{\bar{\imath}_j,1} - t) \geq w_2/(U_{\bar{\imath}_j,2} - t) \geq \cdots\}$" with "loop($q \in J^c_{\bar{\imath}_j} | st_q \geq t$): $\{w_1 \geq w_2 \geq \cdots\}$".

- **Scheduling Heuristic 5:**

  $w_1 \geq w_2 \geq \cdots$.

- **Scheduling Heuristic 6:**

  $w_1(t - L_{\bar{\imath}_j,1}) \geq w_2(t - L_{\bar{\imath}_j,2}) \geq \cdots$.

- **Scheduling Heuristic 7:**

  $w_1/s_{\bar{\imath}_j,1,t} \geq w_2/s_{\bar{\imath}_j,2,t} \geq \cdots$. This criterion is called weighted shortest processing time (WSPT) in the scheduling literature.

In Heuristic 5, we order the jobs according to their weights, since the objective is to minimize the weighted total number of abandonments. Heuristic 6 gives priority to jobs with larger weights and longer waiting time. The intuition behind this algorithm is that a late deadline does not always mean that the ice floe is not a real threat to the system. For instance, it is possible that the deadline for the ice floe is far away, but the ice floe has been in the neighborhood of the alert zone for a long time. In this case, it may be a good idea to serve this ice floe as soon as possible. Heuristic 7 represents the WSPT heuristic, which is the most popular heuristic to solve the problem of minimizing weighted total number of tardy jobs in the scheduling literature.

The computational performance of these scheduling heuristics is presented in the following Chapter. Since model (3.2) provide lower bounds for the multi-stage model and the scheduling heuristics provide upper bounds, our computational experience with the optimality gap is also discussed in the following Chapter.

Chapter 4

Computational Analysis

In this chapter, we discuss our computational experience with the optimization models and the scheduling heuristics presented in Chapter 3. We start with the parameter values for the optimization models, followed by the strategies to improve the solution time. Section 4.3 presents an analysis on the optimality gaps since the two-stage stochastic facility location problem and the scheduling heuristics provide respective lower bounds and upper bounds on the optimal value z^* of the original multi-stage model. The computational performance of the scheduling heuristics is also studied in Section 4.4. Section 4.5 further demonstrates the value of our approach by comparing its performance with a simpler facility location model that fails to fully capture queueing dynamics.

4.1 Input Parameters

Using current technology, satellite images are updated every four hours, and we call this a stage in model (3.1). Increasing the number of stages increases the fidelity of the solutions since the design decisions affect the system over a long horizon, but the model becomes computationally intractable even

when the number of stages is relatively small. That is why we need to develop the two-stage model and the scheduling heuristics to obtain upper and lower bounds for the optimal solution. We use the arrival process simulation methods introduced in Section 2.3.1 to generate synthetic data in order to test the performance of these approximation models.

The parameter θ generates the pair $(R\sin\theta, R\cos\theta)$ on the circumference of the alert zone circle, where R is the radius of the alert zone. Here we assume $\theta \sim Unif[-\frac{\pi}{2}, \frac{\pi}{2}]$, since the platform is close to the continent in reality as shown in Figure 4.1. Parameter $\alpha \in [0, \pi]$ determines the direction of the velocity and we assume $\alpha \sim Unif[0, \pi]$, independent of θ, since we only consider the ice floes that enter the alert zone. The distribution of θ and α depends on the characteristics governing the movement of ice floes, and here we adopt independent uniform distributions for both.

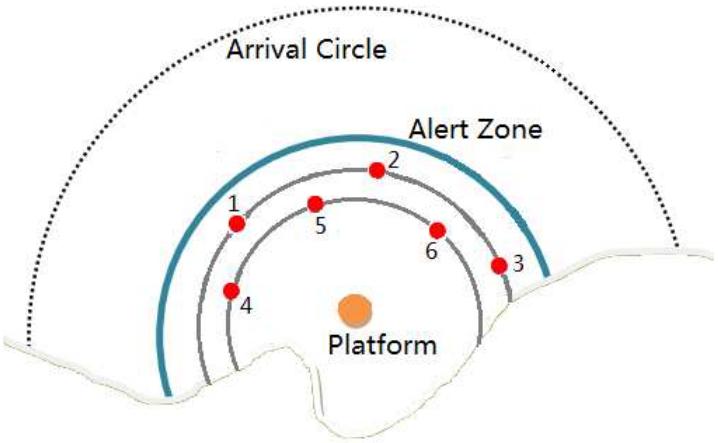

Figure 4.1: Potential docking station locations on two concentric circles.

We first provide a two-stage model with six potential docking station locations on two concentric circles as shown in Figure 4.1. Again, the alert zone radius R is 36 nmi. The outer docking station circle radius r equals to 30 nmi, and the inner docking station circle radius $\tilde{r}$ equals to 25 nmi. During the computational experiments, the locations on the inner circle are never selected as optimal system design decisions to model (3.2), unless there are more than three docking stations to locate. This observation is consistent with intuition since locating docking stations on the outer circle results in shorter service times, and the observation is further consistent with our results from Section 3.3, which suggest the optimal radius of the docking station circle is close to the radius of the alert zone. This analysis suggests that we place the potential docking station locations on one circle instead of several circles, so we assume there are six potential docking station locations as shown in Figure 4.2. The radius r for the docking station circle is set to be 30 nmi by taking both the ocean depth (in a particular case study with industry) and the results from the simplified docking station location model in Section 3.3 into consideration.

We assume the ice floe scan time is uniformly distributed between one hour and eight hours. Also, the AUV speed is assumed to be $v_a = 8$ knots and the ice floe speed is $v = 0.5$ knots. We assume the AUV battery life is 48 hours, and the AUVs must charge for four hours after each scanning sortie. Due to limited capacity at the docking station, we assume at most six AUVs can be allocated to each docking station. We generate scenarios using a spatial

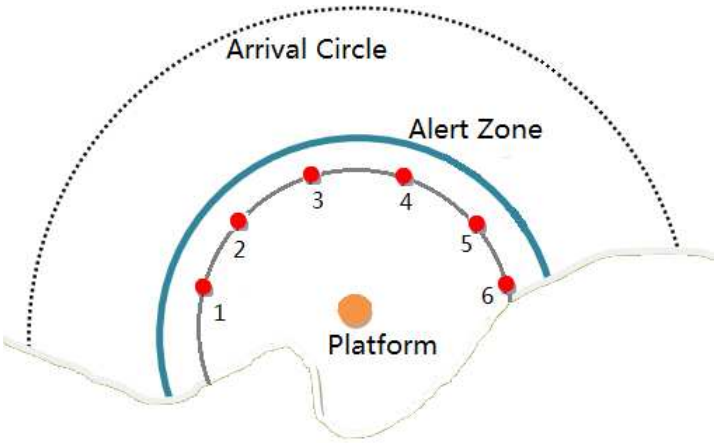

Figure 4.2: Potential docking station locations on one circle.

Poisson process with $\lambda = 0.5$ arrivals per hour, which means on average one ice floe arrives to the system every two hours. In the following analysis, the probabilities for all the scenarios are equal.

In Section 3.4, we present a two-stage stochastic facility location model called model (3.2) which provides a lower bound on the optimal value of model (3.1). Ideally, we want the optimization model to be large enough so it can provide us with a high fidelity solution. Also, we want it to be computationally tractable. In this section, we explore the appropriate basic time unit (u_q), the number of scenarios ($|\Omega|$) and the length of time horizon (T) for model (3.2). Lemma 1 states that increasing the basic time unit, u_q, decreases the optimal value of model (3.2) if we fix the system design decisions. Assume we put one docking station and four AUVs at location 3 in Figure 4.2. Then, Figure 4.3 shows the (normalized) optimal value, $\hat{z}_q^*$, under

three independent and identically distributed (i.i.d.) data sets with different values of u_q.

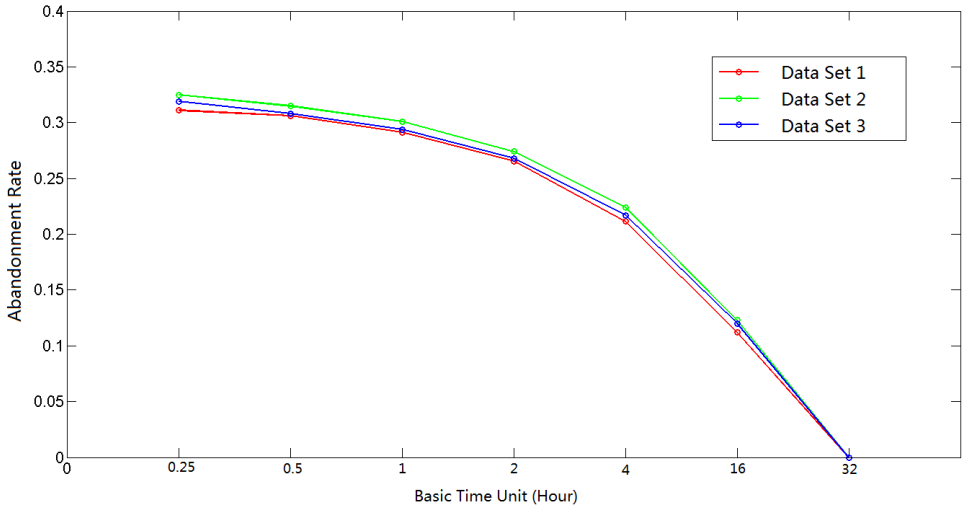

Figure 4.3: Normalized optimal value from two-stage model (3.2) with fixed design decisions ($y_3 = 1$, $x_{3,0} = 4$; $y_i = 0$, $x_{i,0} = 0$, $\forall i \neq 3$, see Figure 4.2). The y-axis corresponds to $\hat{z}_q^*/\sum_{\omega\in\Omega} p^\omega |J^\omega|$, the normalized optimal value of model (3.2). The x-axis corresponds to u_q, the basic time unit.

Figure 4.3 plots the abandonment rate, $\hat{z}_q^*/\sum_{\omega\in\Omega} p^\omega |J^\omega|$, instead of the expected number of abandonments to normalize the model output. The time horizon $T = 512$ hours, and we generate $|\Omega| = 5$ i.i.d. scenarios for each model instance. Figure 4.3 shows the abandonment rate decreases as the basic time unit u_q increases, and the rate drops more quickly as u_q grows. These observations are consistent with Lemma 1, since optimistic rounding has a greater impact on model (3.2) when u_q is large. For example, when u_q is 32 hours, all the ice floes with service times less than 32 hours can be served

instantaneously in model (3.2) due to the optimistic rounding, and so the abandonment rate drops to zero in all the data sets shown in Figure 4.3. On the other hand, the abandonment rates only change slightly when u_q decreases from 0.5 hour to 0.25 hour, but the required computational effort to solve model (3.2) to optimality grows from two hours to five hours. All computations we report here are carried out on a 3.33 GHz Xeon processor with 24 GB of shared memory, using CPLEX version 12.4.

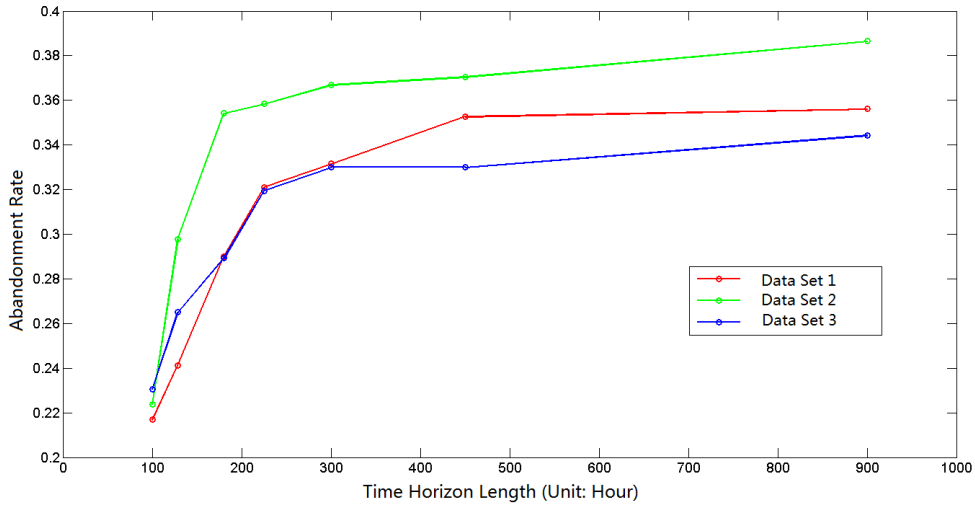

Figure 4.4: Results from Heuristic 2 with different time horizon T.

Another issue is the tradeoff between the time horizon, T, and the number of scenarios, $|\Omega|$. For example, we can have one long scenario, or several short scenarios for each model. Assume we use Heuristic 2 with the same design decisions $y_3 = 1$ and $x_{3,0} = 4$. Figure 4.4 shows the relation between the abandonment rate and T for three i.i.d. data sets. Here we start

with one scenario with $T = 900$ hours. Then common random numbers are used to generate two scenarios with $T = 450$ hours for each scenario. Following the same technique, we generate up to nine scenarios with $T = 100$ hours for each scenario. All AUVs are available at the beginning of each scenario, which introduces *initialization bias*. Also, those ice floes whose time window upper bounds exceed T are ignored, and we call this *cutoff bias*. Decreasing T increases both the initialization bias and the cutoff bias, which lowers the abandonment rate. From Figure 4.4 we can see the abandonment rate is relatively stable when T exceeds 225 hours, and it decreases dramatically for smaller values of T.

Based on these results, we choose $T = 250$ hours, and, $u_q = 0.5$ hours to evaluate the optimality gap using $|\Omega| = 2$ i.i.d. scenarios for each data set. Figure 4.5 shows optimal results from model (3.2) when we change the number of stations and the number of AUVs. We can see that the percentage of ice floes we fail to scan, which is the abandonment rate, drops quickly as we increase the number of AUVs. When the number of AUVs is fixed, increasing the number of docking stations also decreases the abandonment rate. For example, when we have four AUVs and one single docking station, the ice floe abandonment rate is 27.36%. This value drops to 21.63% and then to 19.51% as we allocate four AUVs to two, and then three, docking stations.

We use CPLEX as the solver for our mixed integer optimization model. In the next section, we discuss the strategies to reduce the solution time of model (3.2).

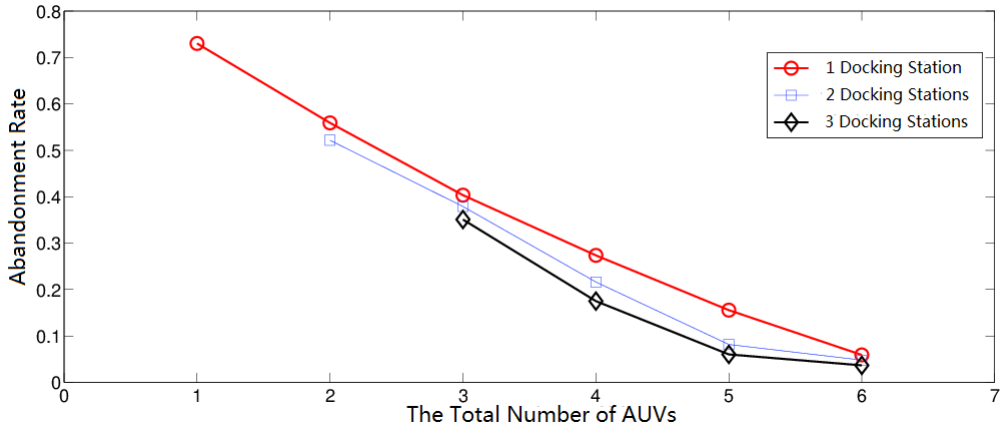

Figure 4.5: Efficient frontier depicting the tradeoff between the abandonment rate and the total number of AUVs.

4.2 Strategies to Reduce MIP Solution Time

We use the General Algebraic Modeling System (GAMS, version 23.9.1) to implement our mixed integer programming (MIP) models. GAMS links to a number of high performance solvers and we use IBM CPLEX (version 12.4) here. All the computational experiments regarding MIP optimization are performed on the University of Texas at Austin's Mechanical Engineering high performance computing machines, where "the machines each have two sixcore, hyperthreading 3.33 GHz Xeon processors and 24 GB of shared memory." In this section, we discuss the strategies to improve CPLEX performance, and thus to reduce the solution time of the model. We refer to Klotz and Newman [2] for more general guidelines to solve difficult MIP models.

In order to solve a MIP, CPLEX uses a branch-and-bound algorithm, which is an intelligent enumeration method, along with cut generation. The

branch-and-bound process starts with the linear programming (LP) relaxation of the MIP, where it assumes continuous values for all the decision variables. The solution obtained from the LP relaxation is used as a starting point for the branch-and-bound algorithm to obtain an optimal solution for the MIP. In our spatial detection model, the objective is to minimize the expected total number of abandonments. So the LP relaxation provides a lower bound for the MIP, since the feasible region of the MIP is a subset of the feasible region for the LP relaxation.

If the LP relaxation is infeasible, then the MIP is also infeasible. If the optimal solution from the LP relaxation is feasible for the MIP, then the solution is also optimal for the MIP. A solution with all the decision variables equal to zero is feasible for both model (3.1) and model (3.2), which means that the MIP models and the LP relaxations are always feasible in this case. But according to our computational experiments, the optimal solution from the LP relaxation is usually not feasible for the MIP, which means branch-and-bound process has to be employed to arrive at an optimal solution for model (3.2).

Without loss of generality, we denote the LP relaxation of a model in standard form as model (P), and we also label its dual problem as model (D):

$$(P) \quad \min \quad c^T x \tag{4.1a}$$

$$\text{s.t.} \quad Ax = b \tag{4.1b}$$

$$x \geq 0; \tag{4.1c}$$

$$(D) \quad \max \quad y^T b \tag{4.2a}$$

$$\text{s.t.} \quad y^T A \leq c^T. \tag{4.2b}$$

See Chapter 4 of Bertsimas and Tsitsiklis [12] for more on duality theory. There are different LP solution algorithms implemented in CPLEX, and a user can specify the LP algorithm for both the root node and the child nodes of the branch-and-bound tree [1]. If left to its default, CPLEX automatically chooses the LP algorithm for the model. The three most popular LP algorithms are the primal simplex algorithm, dual simplex algorithm, and the interior point algorithm. We can apply either the primal simplex algorithm or the dual simplex algorithm on both model (P) and model (D) given below, and so there are four different ways to utilize the simplex algorithm on the LP relaxation.

Let A denote an $m \times n$ matrix. Then, model (P) has m constraints and n decision variables. The simplex algorithm moves from one vertex of the feasible region to another with improved objective function value, until an optimal solution is reached. So the simplex algorithm terminates with an optimal basis for the LP relaxation and the branch-and-bound process uses it as a starting point for the enumeration. For highly degenerate [21] LP models, the simplex algorithm may not be the best option since it may cycle among, or stall temporarily at, solutions with the the same objective value.

Unlike the simplex algorithm, an interior point algorithm moves

through the feasible region of the LP relaxation to find an optimal solution, which means it will not provide a basic feasible solution if the model has multiple optimal solutions. One of the difficulties during the implementation of an interior point algorithm is to construct an optimal basis, and CPLEX uses a crossover algorithm (see, e.g., [11]) for that. However, a crossover algorithm can be computationally expensive, and so it may not be a good idea to employ the interior point as the LP algorithm for the root nodes.

In model (3.2), we have more decision variables than constraints, which means $m \ll n$ for our matrix A. For this type of LP, applying the primal simplex or the interior point algorithm on the primal problem may be more efficient. By using the simplex algorithm, the number of constraints m has more impact on the model solution time since it uses an $m \times m$ basis matrix at each iteration. Typically, larger values of m indicate longer model solution times, and so working on model (P) is more desirable in this case. Also, the primal simplex may have more advantages than the dual simplex since the primal simplex only needs to calculate the reduced costs for a small subset of nonbasic variables at each iteration in this case. When $m \ll n$, the interior point algorithm is likely to be effective since AA^T has relatively small size (an $m \times m$ matrix). We refer to Klotz and Newman [1] for a detailed comparison of different LP algorithms. For model (3.2), if we leave CPLEX to its default setting, it chooses dual simplex on the primal problem to solve the LP for the root node. When we change the root node LP algorithm to interior point, CPLEX solves the LP relaxation much faster. For the child nodes, dual simplex

on the primal problem works best for our model.

We can give the MIP an initial feasible solution, which may also help to improve the solution time. In model (3.2), it may be challenging to come up with an initial feasible solution which includes both the system design decisions and the operation decisions. However, it is straightforward to come up with a partial initial feasible solution which only has the system design decisions, including the docking station locations and the AUV allocations. We can provide the partial initial feasible solution to CPLEX to help reduce the solution time.

In model (3.2), the integer decision variable $x_{i,t}^{\omega}$ indicates the total number of available AUVs at location i at time t in scenario ω. Relaxing $x_{i,t}^{\omega}$ to be continuous does not affect the optimal solution of the model as long as the decision variable $x_{i,0}$ is integer and $z_{i,j,t}^{w}$ is binary, since the inventory constraints ensure $x_{i,t}^{\omega}$ to be integer in this case. While changing a pure IP problem into a MIP problem does not always help to reduce the solution time, relaxing $x_{i,t}^{\omega}$ generally helps model (3.2) to arrive at an optimal solution more quickly.

CPLEX allows users to set their preferences for cut generation. Generally, aggressively generating cuts introduces extra constraints that may not be helpful, which can increase solution time. When left as the default, CPLEX generates cuts only if they seem to be useful. There are different types of cuts that can be generated, e.g., clique cuts, mixed integer rounding cuts, and cover cuts. During the branch-and-bound process for our MIP models, we

notice that generating symmetry cuts aggressively usually helps the algorithm to arrive at an optimal solution more quickly. This suggests that the model has intrinsic symmetry that needs to be removed. When the branch-and-bound tree has some symmetric nodes, it is more difficult to prune what amount to redundant branches, and then the optimizer has to keep track of more nodes, which increases the overall solution time of the model.

Besides generating symmetry cuts, there are several other methods that may help to remove intrinsic symmetry in a model. The original objective function of our spatial detection model is to minimize the expected total number of abandonments, where we assign weight p^ω to scenario ω. This means the ice floes that arrive in the same scenario have the same weights, which may introduce symmetry. Instead, we can assign weight q_j^ω to ice floe j in scenario ω, so different ice floes have different weights. The weight q_j^ω can be assigned based on small perturbations, e.g., $q_j^\omega \sim Unif[1, 1+\zeta]$, $\forall j \in J^\omega, \omega \in \Omega$, where ζ is a very small number. We can also assign the weight according to different characteristics of the ice floe, e.g., the size of the ice floe, the distance from the ice floe to the platform, or the speed of the ice floe. The new objective function is shown by (4.3):

$$\sum_{\omega \in \Omega} \sum_{j \in J^\omega} q_j^\omega \cdot (1 - r_j^\omega). \tag{4.3}$$

Another way to break intrinsic symmetry is to introduce a secondary objective function that distinguishes the ice floes. For example, we can denote the service completion time as sc_j^ω for ice floe j in scenario ω and use it as

a secondary objective function. Then, the new objective function is defined by (4.4) and we assign weight ϵ for the secondary objective:

$$\sum_{\omega\in\Omega} p^{\omega} \sum_{j\in J^{\omega}} (1 - r_j^{\omega}) + \epsilon \cdot \sum_{\omega\in\Omega} \sum_{j\in J^{\omega}} sc_j^{\omega}. \tag{4.4}$$

With the original objective function, model (3.2) provides lower bounds for the multi-stage model according to Theorem 2. We can prove that if we change the objective function in both the multi-stage model and the two-stage model as (4.3) or (4.4), then the optimal objective value of the two-stage model is still a lower bound for the multi-stage model.

When using CPLEX as the solver for a MIP, another important way to control the branch-and-bound efficiency is to define branching priorities for the decision variables. When left to its default, CPLEX makes the decision about which variable to branch on at each node. If we assign different priorities to different decision variables, then CPLEX branches on the higher priority variables before the lower priority variables. In model (3.2), we can assign higher priority for the design decisions, which includes the docking station locations and the AUV allocations at each station, and lower priority for the system operation decisions.

Suspected Issues	Strategies	Average Performance Improvement (based on the original model)

Table 4.1: Strategies to improve MIP solution time for model (3.2).

Inefficient root node LP algorithm	Apply interior point algorithm on primal	Root node LP solution time decreases 95%
Inefficient child node LP algorithm	Apply dual simplex algorithm on primal	Overall MIP solution time stays the same
Lack of initial feasible solution	Assign a partial initial feasible solution	Overall MIP solution time increases 20%
Unnecessary integer decision variables	Relax the decision variable $x_{i,t}^{\omega}$	Overall MIP solution time decreases 15%
Intrinsic symmetry	Generate symmetry cuts aggressively	Overall MIP solution time decreases 6%
	Assign different weights to different ice floe arrivals	MIP solution time stays the same for small weight variation; MIP solution time increases when weight variation increases
	Add a secondary objective function to minimize the sum of the service completion times	Overall MIP solution time decreases up to 90% when the weight for the secondary objective function increases
Inefficient priority branching	Assign higher priority for design decisions and lower priority for operation decisions	Overall MIP solution time stays the same
Inefficient cut generation	Generate cuts aggressively, e.g., clique, cover, zero-half and mixed integer rounding	Overall MIP solution time increases

Table 4.1: Table 4.1 cont.

Table 4.1 shows a summary for all the strategies we discuss in this

section to improve the solution time of model (3.2). The first column has a short description for each of the suspected issues, and the second column contains the proposed solutions for each specific issue. The third column shows the average performance improvement based on the original model, where all the CPLEX parameters are set as the default. We apply these techniques on five instances of our MIP; each instance has 500 time periods and two i.i.d. scenarios. We adopt the same assumptions on the other parameter settings as introduced in the previous section. When we apply interior point algorithm on the primal, relax the decision variable $x_{i,t}^{\omega}$ and generate symmetry cuts aggressively in CPLEX, the overall solution time for model (3.2) decreases 17% on average.

CPLEX chooses dual simplex to solve the primal LP for the root node, and we can see great improvement of the solution time for the LP relaxation when we instead use an interior point algorithm at the root node. On average, the root node LP solution time reduces 95% and the overall MIP solution time reduces 4%. Also, the crossover process only takes a few seconds to construct an optimal basis. For the child nodes, CPLEX uses dual simplex on the primal, which is also the best algorithm based on our computational experiments. When we assign a partial initial feasible solution to the MIP, the solution time for most of the instances grows, indicating a partial initial feasible solution does not help to improve the solution time of our MIP in general. One possible reason is that CPLEX turns off some of its heuristics when a partial initial feasible solution is provided, while these heuristics help

the algorithm arrive at an optimal solution more quickly. When we relax the decision variables $x_{i,t}^{\omega}$ to be continuous, the solution time decreases 15% on average.

To eliminate intrinsic symmetry in our model, we develop three different strategies. Generating symmetry cuts aggressively for model (3.2) helps to reduce the solution time around 6% on average. When we change the objective function of the MIP, it becomes a different model and the optimal solutions may also change. When we assign different weights to different ice floes, we start with small weight variation and assume $q_j^{\omega} \sim U[1, 1.01]$. The solution times of the MIP stay the same, and the optimal solutions are the same as the original model. When we increase the weight variation, which means increasing the upper bound of the uniform distribution for q_j^{ω} in this case, the average MIP solution time increases.

When a secondary objective is introduced, the weight, ϵ, for the secondary objective function should be chosen judiciously according to the magnitude of the primary objective. For model (3.2), the overall solution times stay the same when the weight ϵ is very small, e.g., $\epsilon = 0.0001$ in this case. When we increase the value of ϵ, the solution times decrease, e.g., when we set $\epsilon = 0.001$, it decreases 58% on average. The solution time for model (3.2) decreases up to 90% if we keep increasing the value of the weight ϵ. There is no improvement in the overall solution time when we manually assign higher priority for the system design decisions and lower priority for the operation decisions.

Besides generating symmetry cuts, we can also try to generate other types of cuts for the model more aggressively. Clique cuts help to distinguish incompatible variables, e.g., two binary variables are called incompatible if the model has a constraint that specifies the sum of these two variables cannot exceed one. The mixed integer rounding (MIR) cuts create cutting planes that truncate fractional bounds for integer variables. Generating cover cuts are usually helpful when the model has 0-1 knapsack constraints. Zero-half cuts can be generated when the left-hand side of a constraint consists of integer decision variables and integer parameters, while the right-hand side has fractional values. Our computational experiments suggest we should leave these cut generations as default in CPLEX, since CPLEX only seems to generate the relevant ones. Manually forcing the optimizer to generate cuts aggressively introduces unnecessary constraints in our model, and increases the MIP solution time.

We solve model (3.2) to optimality in this dissertation, but it is worth mentioning that the solution time reduces greatly if we allow a reasonable relative tolerance when we solve model (3.2). For example, if the relative tolerance is 5%, the MIP solution time reduces about 60% on average.

In the next section, we show the optimality gaps provided by the two-stage optimization model (3.2) and the online scheduling heuristics from Chapter 3. All the useful strategies we discuss in this section are applied to the MIP to reduce its solution time.

4.3 Optimality Gap

According to our study of the effect of selecting certain parameters in Section 4.1, we choose $T = 250$ hours and $u_q = 0.5$ hours to evaluate the optimality gap using $|\Omega| = 2$ i.i.d. scenarios in each data set. On average, it takes eight hours to solve one instance of model (3.2) in CPLEX with these parameters. Note there here we are optimizing both design and operation decisions while the results we report in Section 4.1 use fixed design decisions. Different traffic intensity levels are taken into consideration since the quality of the optimality gap differs as the traffic intensity level changes. Similar to the analysis of the simulation results in Chapter 2, we define the high traffic intensity level to have abandonment rates above 40%, while the medium level has abandonment rates between 10% and 40%, and the low traffic intensity level has abandonment rates below 10%. The optimal design decisions obtained from model (3.2) are used as the system design decisions in the scheduling heuristics.

Figure 4.6 shows the optimality gap for five i.i.d. data sets with a single docking station, by plotting the upper bound from a heuristic and the lower bound from model (3.2)'s optimal value. We change the traffic intensity by adjusting the total number of AUVs, from $h = 2$ to $h = 4$ to $h = 6$. Since the objective is to minimize the expected number of ice floe abandonments, Heuristics 1, 2 and 3 generate upper bounds, and we choose the tightest one to report in Figure 4.6. With $h = 2$ AUVs, the minimum relative gap is 1.5% and the average is 3.1%, while the minimum absolute gap is 0.9% and the

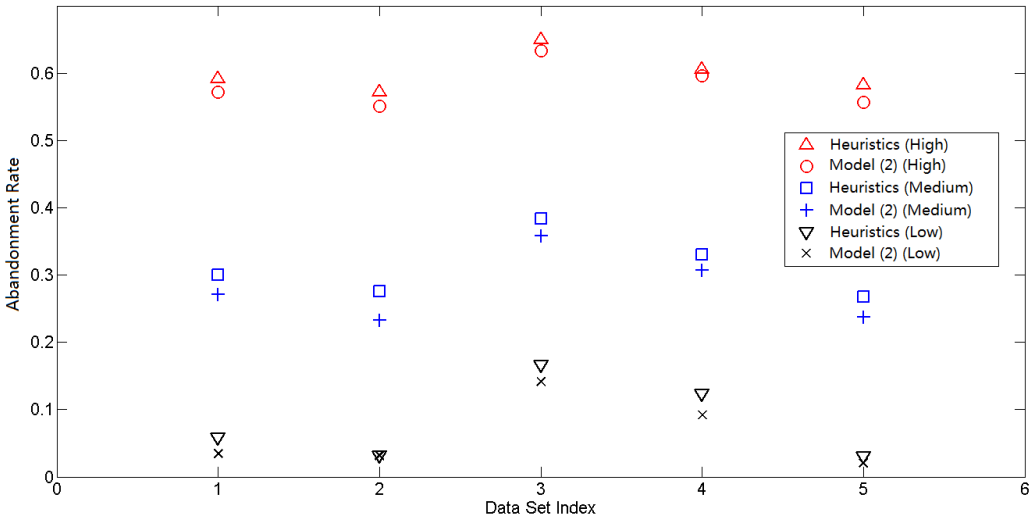

Figure 4.6: Optimality gaps with one docking station.

average is 1.8%. When we increase the AUV quantity to $h = 4$, the minimum relative gap increases to 6.5% and the average increases to 10%. Also, the minimum absolute gap becomes 2.30% and the average is 3%. If we increase the AUV quantity to $h = 6$, the lower bound provided by model (3.2) equals to the upper bound obtained from Heuristic 3 in data set 2, so a zero gap is achieved in this case. However, the average relative gap increases to 23.1% and the average absolute gap becomes 1.8%. Here, the absolute gap for the abandonment rate is the difference between the lower bound, $z_q^*/\sum_{\omega\in\Omega} p^\omega|J^\omega|$, and the upper bound obtained from the heuristics. The relative gap is the same difference divided by $z_q^*/\sum_{\omega\in\Omega} p^\omega|J^\omega|$.

Table 4.2, Figure 4.7 and Figure 4.8 show the optimality gap statistics from 20 i.i.d. data sets under different traffic intensities and different numbers of docking stations. Table 4.2 shows the average, the minimum, and the

maximum for both the absolute gap and the relative gap. For example, for a single station with high traffic intensity, the average absolute gap is 2.02%, while the minimum absolute gap is 0.59%, and the maximum is 3.04%.

Single Station: Traffic Intensity	Absolute Gap	Relative Gap
High	2.02% [0.59%, 3.04%]	3.18% [0.78%, 5.05%]
Medium	3.29% [2.17%, 4.52%]	10.27% [4.65%, 18.64%]
Low	2.19% [0, 3.74%]	21.92% [0, 41.66%]
Multi-Station: Traffic Intensity	**Absolute Gap**	**Relative Gap**
High	3.67% [2.78%, 5.32%]	6.19% [4.69%, 9.43%]
Medium	6.74% [4.74%, 8.97%]	17.06% [10%, 26.09%]
Low	6.76% [2.67%, 9.91%]	39.5% [27.27%, 55.17%]

Table 4.2: Statistics of the optimality gaps. Here $T = 250$ hours, $|\Omega| = 2$, and $u_q = 0.5$ hours for model (3.2). We change the AUV number h from 2 to 6 to achieve different traffic intensity levels, and the docking station number $m = 2$ or $m = 3$ in the multi-station case.

For the single station case, the bounds are reasonably tight when the traffic intensity is high since the average relative gap is 3.18% and the minimum relative gap is 0.78%. When we increase the total number of AUVs, which implies a decrease in the traffic intensity, the absolute gap stays relatively stable. For example, the average absolute gap is 2.02% when the traffic intensity is high, 3.29% for medium traffic intensity, and 2.19% for low traffic intensity. The abandonment rate decreases as we increase the total number of AUVs, so the average relative gap increases to 10.27% for medium

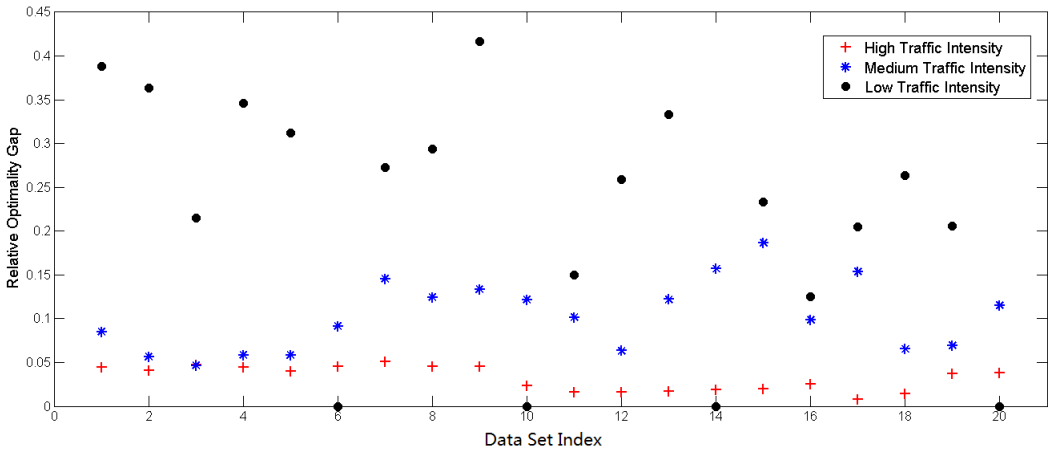

Figure 4.7: Relative optimality gaps for one docking station.

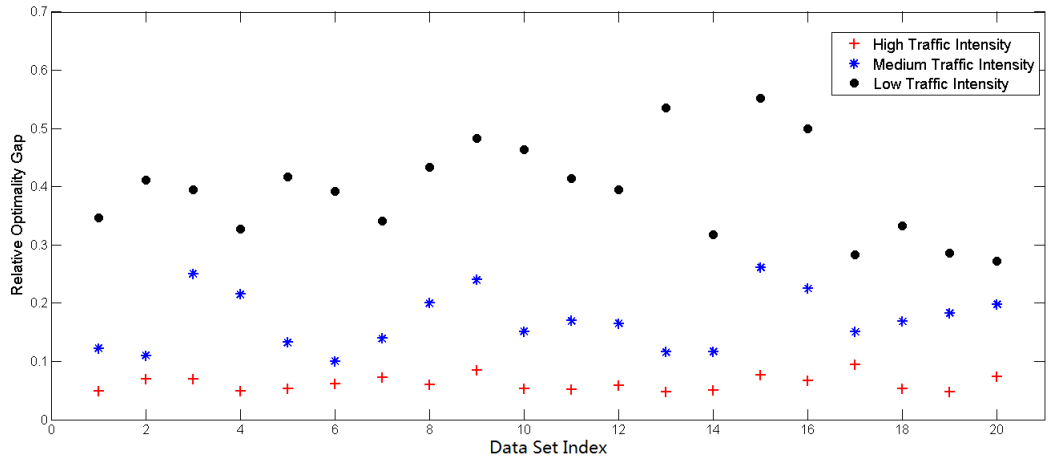

Figure 4.8: Relative optimality gaps for multiple docking stations.

traffic intensity and 21.92% for low traffic intensity. Even though the average optimality gap is not ideal when traffic intensity is low, Figure 4.7 shows that a zero optimality gap is achieved four times in this case, which indicates the optimal solutions for model (3.1) are obtained. When more docking stations are added to the system, both the absolute optimality gap and the relative

optimality gap increase, as shown by Table 4.2 and Figure 4.8. One important reason is that model (3.2) can capture the optimal collaboration dynamically between docking stations as an optimization model, while both scheduling heuristics rely on predetermined rules to decide the station collaboration. With multiple docking stations, the average relative gap is 6.19% for high traffic intensity, 17.06% for medium traffic intensity, and 39.50% when traffic intensity is low.

During the experiments, we find that Heuristic 2 and Heuristic 3 always outperforms Heuristic 1. Heuristic 2 generally performs better when the abandonment rate is higher than 10%, while Heuristic 3 has better performance otherwise. The detailed computational performance of all the scheduling heuristics is introduced in the following section.

4.4 Performance of the Scheduling Heuristics

For the scheduling problem of minimizing the expected number of abandonments, we develop three heuristic algorithms in Section 3.5.1. Here we assume the length of the time horizon $T = 720$ hours and the docking station locations are the same for each heuristic. We generate 14 i.i.d. data sets and change the traffic intensity by changing the total number of AUVs. Figure 4.9 shows the computational performance for Heuristics 1, 2, and 3. The y axis shows the value of the objective and the x axis indicates the data set index. We can see that Heuristic 2 always has lower abandonment rate than Heuristic 1. When the abandonment rate exceeds 10%, Heuristic 2 provides

the lowest abandonment rate. When the abandonment rate is lower than 10%, Heuristic 3 achieves the best performance. This indicates we should choose Heuristic 2 under high and medium traffic intensities, and Heuristic 3 under low traffic intensity as the scheduling heuristic for our spatial detection model in order to minimize the expected number of ice floe abandonments.

Heuristic 1 is similar to the FIFO queueing policy that we use in the simulation model, since here we always abandon the ice floes that cannot be finished before their due dates. Heuristic 2 instead abandons the ice floe with the longest service time, and Heuristic 3 is similar to the EDD queueing policy in the simulation model analyzed in Section 2.4. Both the computational analysis here and the simulation results suggest that FIFO is not a good policy when the objective is to minimize abandonment rate.

Four heuristic algorithms are presented in Section 3.5.2 for the problem of minimizing the weighted total number of abandonments. Here we assume the weight for the ice floes has a uniform distribution between 0.5 and 1. The y axis shows the value of the objective and the x axis indicates the data set index in Figure 4.10. Here we also assume the length of the time horizon is 720 hours and the docking station locations are the same for each heuristic. We again generate 14 i.i.d. data sets and change the traffic intensity by changing the total number of AUVs.

Figure 4.10 shows the computational performance for these four heuristics. We can see that Heuristic 6 only achieves the lowest objective value in the first data set, which indicates that Heuristic 6 generally has

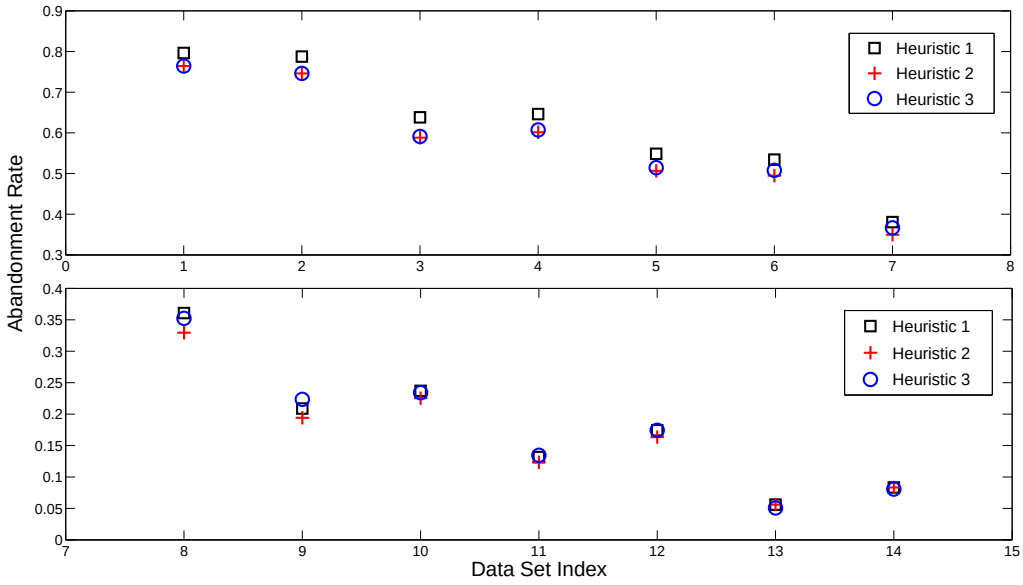

Figure 4.9: Computational performance for Heuristics 1, 2, and 3. The design decisions are fixed: $y_3 = 1$, $y_i = 0$, $x_{i,0} = 0$, $\forall i \neq 3$, and we change $x_{3,0}$ from 1 to 6 to change the traffic intensity; see Figure 4.2 for the potential docking station locations

worse performance than the other heuristics. Heuristic 7 achieves the lowest objective value in five data sets during medium and low traffic intensity, while Heuristic 4 achieves three during high and medium traffic intensity. Similar to Heuristic 7, Heuristic 5 also achieves the lowest objective value in five data sets during medium and low traffic intensity.

Table 4.3 summarizes the performance of all the scheduling heuristics. For the problem of minimizing the expected number of abandonments, Heuristic 2 is recommended during high and medium traffic intensity and Heuristic 3 is recommended when traffic intensity is low. When the ice floes have different weights, Heuristic 4 generally has the best performance during

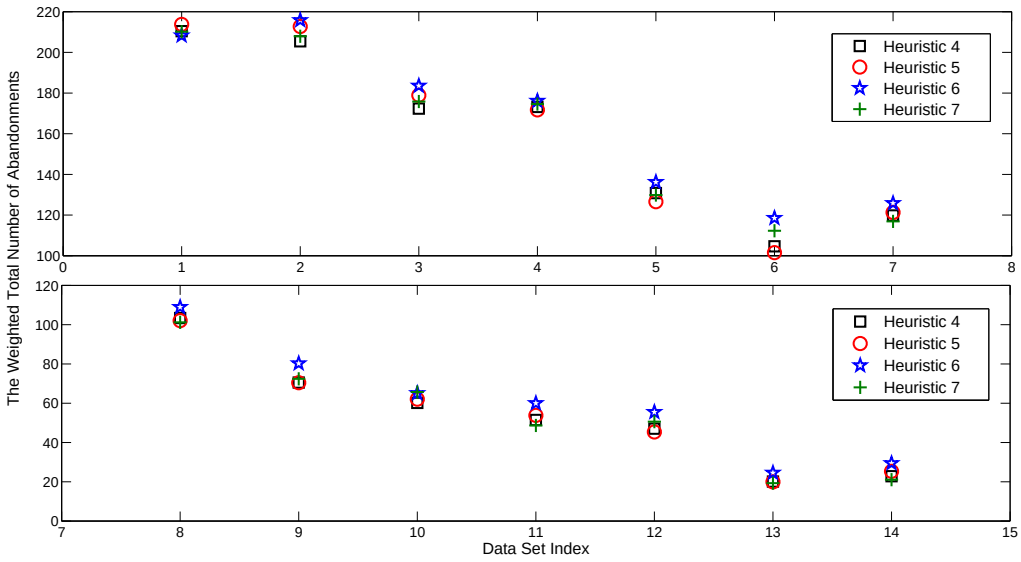

Figure 4.10: Computational performance for Heuristics 4, 5, 6, and 7. The design decisions are fixed: $y_3 = 1$, $y_i = 0$, $x_{i,0} = 0$, $\forall i \neq 3$, and we change $x_{3,0}$ from 1 to 6 to change the traffic intensity; see Figure 4.2 for the potential docking station locations

high traffic intensity, and we can choose either Heuristic 5 or Heuristic 7 during low traffic intensity. For medium traffic intensity, we recommend selecting either Heuristics 4, 5, or 7.

4.5 Facility Location Problem without Queueing Dynamics

Both model (3.1) and model (3.2) have wait-time-dependent service times and inventory constraints that keep track of system queueing dynamics. These two characteristics distinguish our facility location problem from traditional facility location problems. In this section, we introduce a simplified

Traffic Intensity	Minimize the Number of Abandonments
High	Heuristic 2
Medium	Heuristic 2
Low	Heuristic 3

Traffic Intensity	Minimize the Weighted Number of Abandonments
High	Heuristic 4
Medium	Heuristic 4 or Heuristic 5 or Heuristic 7
Low	Heuristic 5 or Heuristic 7

Table 4.3: Scheduling heuristics recommendation.

version of our facility location problem, which does not include the queueing dynamics, and we compare it with model (3.2) to show the importance of incorporating queueing dynamics in our facility location model.

Facility Location Problem without Queueing Dynamics

$$\tilde{z}_q^* = \min \sum_{\omega\in\Omega} p^\omega \sum_{j\in J^\omega} (1 - r_j^\omega) \tag{4.5a}$$

$$\text{s.t.} \sum_{i\in I_j^\omega} \sum_{t\in TW_{i,j,q}^\omega} z_{i,j,t}^\omega = r_j^\omega, j \in J^\omega, \omega \in \Omega \tag{4.5b}$$

$$\sum_{t\in TW_{i,j,q}^\omega} z_{i,j,t}^\omega \le y_i, i \in I_j^\omega, j \in J^\omega, \omega \in \Omega \tag{4.5c}$$

$$\sum_{i\in I} y_i \le m \tag{4.5d}$$

$$x_{i,0} \le n_i y_i, i \in I \tag{4.5e}$$

$$\sum_{i\in I} x_{i,0} \le h \tag{4.5f}$$

$$\sum_{j\in J_i^\omega} \sum_{t\in TW_{i,j,q}^\omega} s_{i,j,t}^\omega z_{i,j,t}^\omega \le (T + \bar{s}_i^\omega) x_{i,0},$$

$$\forall i \in I, \omega \in \Omega \tag{4.5g}$$

$$\sum_{j \in J_i^\omega} \sum_{\tau=\min\{t+u_q,T\}}^{\min\{t+\hat{s}_i^\omega-u_q,T\}} z_{i,j,\tau}^\omega \leq x_{i,0} - \sum_{j \in J_i^\omega} z_{i,j,t}^\omega,$$

$$\forall i \in I, t \in T_q \setminus \{0\}, \omega \in \Omega \tag{4.5h}$$

$$y_i \in \{0,1\}, i \in I \tag{4.5i}$$

$$x_{i,0} \in \{0,1,...,n_i\}, i \in I \tag{4.5j}$$

$$z_{i,j,t}^\omega \in \{0,1\}, i \in I_j^\omega, j \in J^\omega, t \in TW_{i,j,q}^\omega, \omega \in \Omega \tag{4.5k}$$

$$r_j^\omega \in \{0,1\}, j \in J^\omega, \omega \in \Omega. \tag{4.5l}$$

In model (4.5), we define $\bar{s}_i^\omega = \max_{t \in TW_{i,j,q}^\omega, j \in J_i^\omega}\{s_{i,j,t}^\omega\}$ and $\hat{s}_i^\omega = \min_{t \in TW_{i,j,q}^\omega, j \in J_i^\omega}\{s_{i,j,t}^\omega\}$. So $\bar{s}_i^\omega$ is the maximum service time for location i in scenario ω and $\hat{s}_i^\omega$ is the corresponding minimum service time. The optimal objective function value is denoted as $\tilde{z}_q^*$ and constraints (4.5a)-(4.5f) are the same as constraints (3.2a)-(3.2f) in model (3.2). Unlike model (3.2), model (4.5) does not track the AUV inventory, $x_{i,t}^\omega$, dynamically at each time period t. Instead, model (4.5) restricts the AUV usage based on the capacity at a lower resolution through constraints (4.5g) and (4.5h).

Constraint (4.5g) indicates that the total number of time periods required to serve the arrivals at location i cannot exceed the maximum number of time periods available at that location. Here T is the length of the time horizon, and the right-hand side of constraint (4.5g) represents the maximum number of time periods available at location i, since an AUV can be dispatched at $t = T$ and the maximum possible service time is $\bar{s}_i^\omega$. Constraint (4.5h)

restricts excessive dispatch of the AUVs, while the right-hand side is the maximum number of AUVs left at location i at time t and the left-hand side is the total number of AUVs dispatched in the following $\hat{s}_i^\omega - 1$ time periods. Since $\hat{s}_i^\omega$ is the minimum service time for location i, an AUV that is dispatched at time t from location i cannot return to that location in the next $\hat{s}_i^\omega - 1$ time periods.

An ice floe arrival can be served by multiple AUVs in model (4.5) while both model (3.1) and model (3.2) assume that an ice floe can be served by at most one AUV. Similar to model (3.2), model (4.5) also provides a lower bound for the multi-stage model if all the time-related parameters are rounded optimistically; see Lemma 1 for the parameter rounding techniques. We compare the design decisions provided by model (4.5) and model (3.2) by fixing the respective design decisions and running our scheduling heuristics. We use the same parameter settings as in Section 4.3 when we solve these two MIP models, where $T = 250$ hours, $|\Omega| = 2$ i.i.d. scenarios per data set, $u_q = 0.5$ hours, the arrival rate $\lambda = 0.5$ arrivals per hour, and the six candidate locations are shown in Figure 4.2. We assume there are $m = 2$ docking stations, and the AUV quantity $h = 2, 3, 4$, or 5. We fix the design decisions we obtain from both models in Heuristic 2 and compare the abandonment rates to study the quality of the design decisions. Figure 4.11 shows that by using the design decisions given by model (3.2), the scheduling heuristic provides lower abandonment rates, which means we have a tighter upper bound for the multi-stage model. The average relative gap between abandonment rates is 5.84% in Figure 4.11,

and the results indicate that model (3.2), which includes queueing dynamics, gives us better system design decisions than model (4.5).

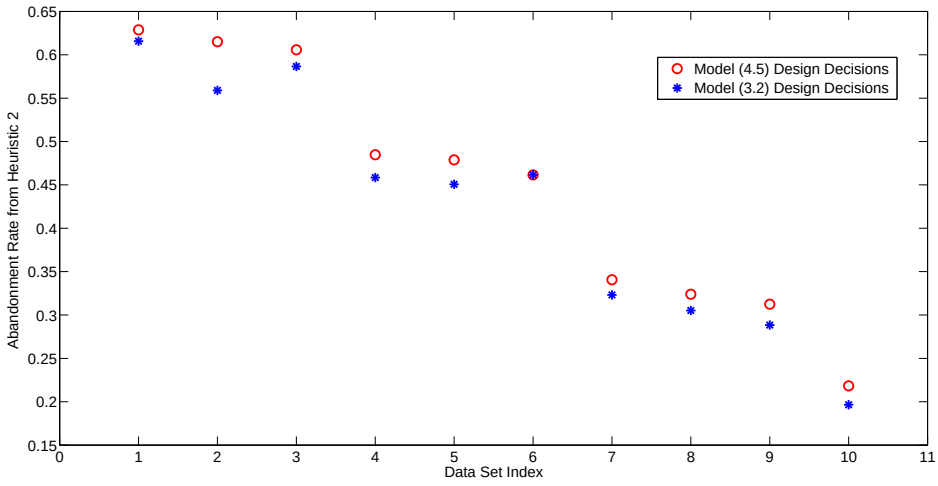

Figure 4.11: Abandonment rates from Heuristic 2 using different design decisions from model (3.2) and model (4.5). Here we generate 10 i.i.d. data sets and change the traffic intensity by changing the AUV quantity h.

Suppose we have seven potential docking station locations instead of six, as shown in Figure 4.12. Assume we have $m = 1$ docking station and $h = 1$ AUV to locate, and again, $\theta \sim Unif[-\frac{\pi}{2}, \frac{\pi}{2}]$ when we generate the arrival trajectories. Due to the symmetry of the arrival trajectories, we know the middle location 4 is the optimal location to place the docking station in the long run. We run model (3.2) and model (4.5) with $m = 1$ and $h = 1$ with four i.i.d. data sets, and Figure 4.13 shows the optimal design decisions from these two models. The green pentagon shows the design decision from model (3.2) while the black circle shows the design decision from model (4.5).

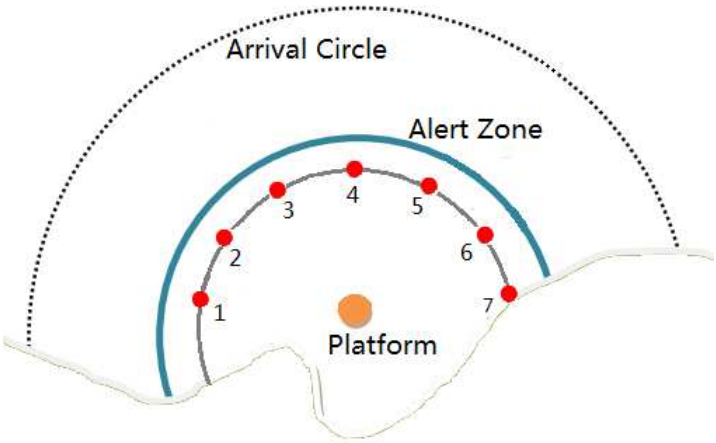

Figure 4.12: Seven potential docking station locations.

We can see that model (3.2) always captures the symmetry in the model by putting the docking station at location 4 in all the data sets, while model (4.5) fails to capture the symmetry in two data sets.

Combining the above two observations, we can see the importance of capturing queueing dynamics in our facility location problem. Another way to check the importance of modeling queueing dynamics is to see the computational behavior of the optimality gap by using the optimal value $\tilde{z}_q^*$ from model (4.5) as a lower bound for the multi-stage model.

We run model (4.5) to obtain a lower bound and a candidate system design decision. Then, we fix the design decision and run the scheduling heuristic to assess the performance of the resulting optimality gap. Figure 4.14 compares the relative optimality gap for both model (3.2) and model (4.5). The

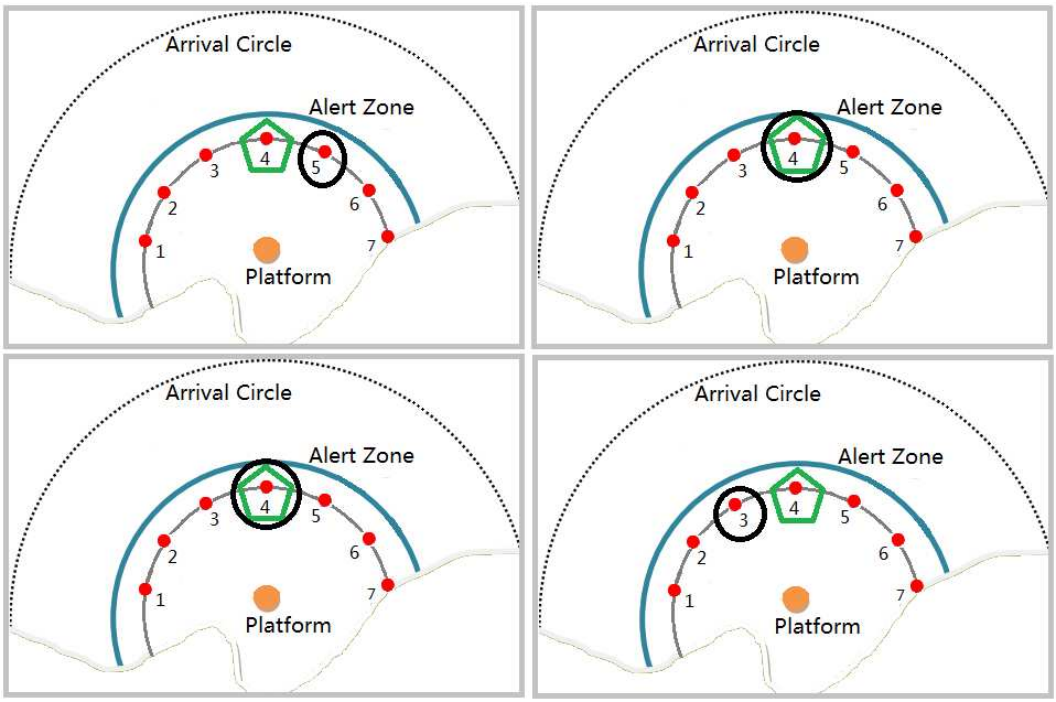

Figure 4.13: The green pentagon shows the design decision from model (3.2) while the black circle shows the design decision from model (4.5) with $m = 1$ docking station and $h = 1$ AUV for four i.i.d. data sets.

gap increases as the traffic intensity decreases in both models, and model (3.2) provides much tighter optimality gap compared with model (4.5). The optimality gap for model (4.5) exceeds 100% under medium traffic intensity and grows to 900% under low traffic intensity. These gaps are dramatically worse than the optimality gaps obtained via model (3.2), as the figure shows. (See also Section 4.3 for a more analysis of model (3.2)'s optimality gap.)

The analysis of this section indicates the importance of capturing queueing dynamics in our facility location model. The facility location problem with queueing dynamics, model (3.2), not only provides better system design

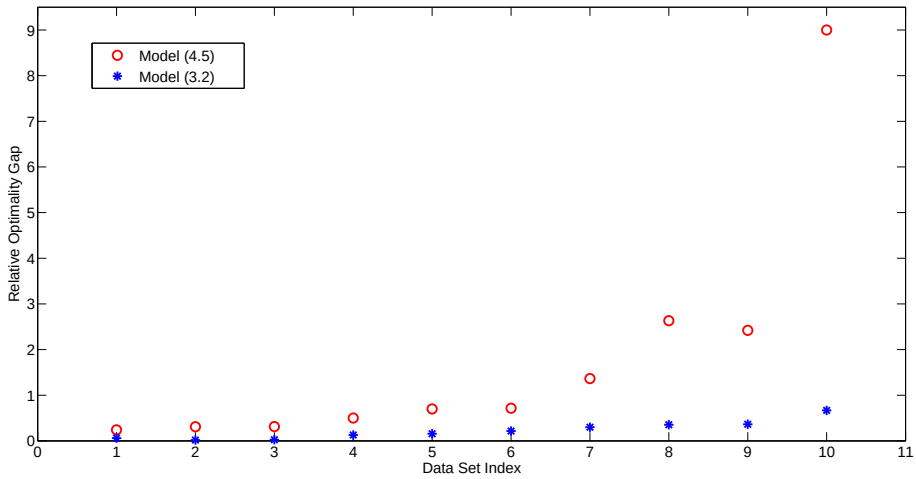

Figure 4.14: Relative optimality gap performance for model (3.2) and model (4.5). Here we generate 10 i.i.d. data sets and change the traffic intensity from high to low by changing the AUV quantity, h. The relative optimality gap increases as the traffic intensity decreases.

decisions, but also provides much tighter optimality gaps when compared with model (4.5), which does not fully capture queueing dynamics.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this dissertation, we develop queueing models, simulation models, and optimization models for a spatial detection system in order to minimize the abandonment rate. The topic arises in the oil industry where drilling platforms in the Arctic Circle need to be protected from ice floe collisions. The system uses AUVs to measure the under-water topography of the ice floes and docking stations to launch and recharge the AUV batteries. There are two types of decisions in this spatial detection model. The design decision includes the locations of the docking stations and the allocations of AUVs at each docking station, while the operation decision involves the scheduling policies of the AUVs.

Even though this model is motivated by an oil industry project, most of the modeling and optimization methods apply broadly to radial detection systems with queueing dynamics. In a general radial detection model, the ice floes are equivalent to the customers arriving to the system, while the AUVs are the servers that provide service for the customers. An abandonment is the same as an unserved customer, so the objective function is to minimize

the rate of unserved customers in a general model. One important aspect of our spatial detection model is that the customer service time depends on the waiting time, and this, coupled with the associated queueing dynamics, distinguishes our model from both existing facility location models and spatial detection models.

In Chapter 2, we develop two queueing approximations and a simulation model to calculate the important system performance measures, e.g., the abandonment probability, the probability of waiting, the expected waiting time and the expected queue length. The $M/M/k+M$ queueing system is used to derive closed form expressions of various performance measures. Also, we propose a water-filling algorithm to allocate AUVs among different docking stations and we prove its optimality if the objective is to minimize the maximum value of function $F(k_i)$, where k_i is the number of AUVs at docking station i and $F(k_i)$ is a non-increasing function of k_i.

The $M/G/k+G$ queue is more complicated than the $M/M/k+M$ queue since closed-form analysis is unavailable, and we rely on approximations to compute the system performance measures. To assess the two queueing approximations, as well as to better study the system dynamics, we develop a discrete event simulation model using the commercial software package Arena. We compare the simulation results with both the $M/M/k+M$ and the $M/G/k+G$ queueing approximation results. Also, different queueing policies are utilized in the simulation model to study the system sensitivity.

Both the design decisions and the operation decisions are taken into

consideration in the optimization models developed in Chapter 3. We model the system as a multi-stage stochastic facility location problem which characterizes the timing of the design decisions, the realizations of randomness, and the operation decisions. Due to the computational complexity of the multi-stage model, several approximation models are developed. A simplified docking station location model is used to provide insights on the potential docking station locations. We also develop a two-stage stochastic facility location problem which provides lower bounds, and several online scheduling heuristics that provide upper bounds for the optimal value of the multi-stage model.

We study the computational performance for the optimization models and the scheduling heuristics, and also evaluate the quality of the optimality gaps provided by the two-stage model and the scheduling heuristics. We choose CPLEX as the optimizer to solve our MIP models and different strategies are discussed in order to decrease the MIP solution time in CPLEX.

5.2 Future Work

Among the queueing models, the $M/G/k + G$ queue provides good approximation during high and medium traffic intensity levels. When the traffic intensity level is low, there is a significant gap in the abandonment probability between the queueing approximation and the simulation results. Different queueing approximation methods can be employed in the future to achieve smaller gaps for the system performance measures at low traffic

intensity. Currently, both the queueing approximations and the simulation model are focused on a single docking station with multiple AUVs. If we want to take all the docking stations into consideration concurrently, a network model is required. One possibility in the future is to develop queueing and simulation network models to obtain more accurate system performance measures.

As we emphasize throughout the dissertation, the customer service time in our spatial detection model depends on the customer's waiting time in queue. In our context this means the service time, $s_{i,j,t}$, for customer j via docking station i depends on the time t. The two-stage stochastic facility location problem provides lower bounds if we round the time-related parameters optimistically, and the change rate of the service time is relatively small, which means $|s_{i,j,t} - s_{i,j,\tau}| \leq |t - \tau|$, $\forall t, \tau$. The corresponding constraints (i.e., those involving $s_{i,j,t}$) amount to resource consumption constraints. A possible direction for future research is to attempt to extend the optimistic rounding theorems of Chapter 3 so that they capture more general time-staged stochastic programs that involve resource consumption constraints. That is, we seek conditions under which we can obtain optimistic bounds by coarsening time and appropriately modifying resource-consumption coefficients.

The two-stage stochastic facility location problem is computationally challenging when the problem size grows large. In order to reduce the model solution time, we discuss various strategies with respect to the optimization engine in Chapter 4. Beyond that, a reformulation of the MIP may lead to

further computational efficiency gains. The computational results in Chapter 4 indicate that the optimality gaps are smaller when we have fewer docking stations in the system. So another possibility is to develop online scheduling heuristics that lead to improved collaborations between different docking stations to achieve smaller optimality gaps. The optimality gap observed between the two-stage model and the scheduling heuristics shows that the variance of the relative gap increases as we decrease the traffic intensity level, so decreasing the variance of the optimality gap is also a potential topic for future work.

Bibliography

[1] E. Klotz and A. M. Newman. Practical guidelines for solving difficult linear programs. *Surveys in Operations Research and Management Science*, 18:1–17, 2013.

[2] E. Klotz and A. M. Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18:18–32, 2013.

[3] G. Koole and A. Mandelbaum. Queueing models of call centers: an introduction. *Annual of Operations Research*, 113:41–59, 2002.

[4] A. Shapiro, D. Dentcheva and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory.* SIAM-Society for Industrial and Applied Mathematics, 1st edition, 2009.

[5] F. Iravani and B. Balcioglu. Approximation for the M/GI/N+GI type call center. *Queueing System: Theory and Application*, 58:137–153, 2008.

[6] D. Gross, J. F. Shortle, J. M. Thompson and C. M. Harris. *Fundamentals of queueing theory.* Wiley, 4th edition, 1998.

[7] F. Lu, J. J. Hasenbein and D. P. Morton. Modeling and optimization of a spatial detection system to minimize abandonment rate. *under review.*

[8] S. A. Khalid, F. Lu, Z. Shu, J. J. Hasenbein and D. P. Morton. Optimization of autonomous vehicles and corresponding docking stations in offshore environment: OTC-24828-MS. Kuala Lumpur, Malaysia, March 2014. Offshore Technology Conference-Asia.

[9] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Macmillan Higher Education, 1st edition, 1979.

[10] N. Mladenovic, J. Brimberg, P. Hansen and J. A. Moreno-Ṕerez. The p-median problem: a survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3):927–939, 2007.

[11] R. E. Marsten, M. J. Saltzman, D. F. Shanno, G. S. Pierce and J. F. Ballintijn. Implementation of a dual affine interior point algorithm for linear programming. *ORSA Journal on Computing*, 1:287–297, 1989.

[12] D. Bertsimas, and J. N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1st edition, 1997.

[13] R. Schultz, L. Stougie and M. H. Vlerk. Two-stage stochastic integer programming: a survey. *Statistica Neerlandica*, 50:404–416, 1996.

[14] J. G. Proakis and M. Salehi. *Digital communications*. McGraw-Hill Science/Engineering/Math, 5th edition, 2007.

[15] W. D. Kelton, R. Sadowski and N. B. Swets. *Simulation with Arena*. McGraw-Hill Education, 5th edition, 2009.

[16] L. V. Green, J. Soares, J. F. Giglio and R. A. Green. Using queueing theory to increase the effectiveness of emergency department provider staffing. *Academic Emergency Medicine*, 13:61–68, 2006.

[17] F. Lu, Z. Shu, J. J. Hasenbein, D. P. Morton, D. P. Berta and S. A. Khalid. Modeling and optimization of docking stations and auvs for ice floe measurement. Banff, Alberta, Canada, September 2012. International Conference and Exhibition on Performance of Ships and Structures in Ice.

[18] J. Rosenhead, M. Elton and S. K. Gupta. Robustness and optimality as criteria for strategic decisions. *Operational Research Quarterly*, 23:413–431, 1972.

[19] M. P. Atkinson and L. M. Wein. Spatial queueing analysis of an interdiction system to protect cities from a nuclear terrorist attack. *Operations Research*, 56:247–254, 2008.

[20] J. D. Banfield and A. E.Raftery. Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. Technical report, University of Washington, Seattle, WA 98195, August 1989.

[21] A. Charnes. Optimality and degeneracy in linear programming. *Econometrica*, 20:160–170, 1952.

[22] H. Heitsch and W. Römisch. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, 118:371–406, 2009.

[23] J. C. Ho and Y. Chang. Minimizing the number of tardy jobs for m parallel machines. *European Journal of Operational Research*, 84:343–355, 1995.

[24] E. L. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977.

[25] A. Mandelbaum and S. Zeltyn. *Advances in services innovations*, chapter Service engineering in action: the Palm/Erlang-A queue, with applications to call centers. Springer, 1st edition, 2006.

[26] A. Molyboha and M. Zabarankin. Stochastic optimization of sensor placement for diver detection. *Operations Research*, 60:292–312, 2012.

[27] S. Nahmias. Perishable inventory theory: a review. *Operations Research*, 30(4):680–708, 1982.

[28] S. H. Owen and M. S. Daskin. Strategic facility location: a review. *European Journl of Operations Research*, 111:423–447, 1998.

[29] M. L. Pinedo. *Scheduling theory, algorithms, and systems*. Springer, 3rd edition, 2008.

[30] J. E. Reed. *Queueing models for large scale call centers*. Dissertation, Georgia Institute of Technology, May 2007.

[31] L. V. Snyder. Facility location under uncertainty: a review. *IIE Transaction*, 38:537–554, 2006.

[32] United States Geological Survey. 90 billion barrels of oil and 1,670 trillion cubic feet of natural gas assessed in the Arctic. Technical report, United States Department of the Interior, Reston, VA, July 2008.

[33] R. Szechtman, M. Kress, K. Lin, and D. Cfir. Model of sensor operations for border surveillance. *Naval Research Logistics*, 55:27–41, 2008.

[34] W. Whitt. Efficiency-driven heavy-traffic approximations for many-server queues with abandonments. *Management Science*, 50:1449–1461, 2004.

[35] W. Whitt. Engineering solution of a basic call center model. *Management Science*, 51:221–235, 2005.

[36] W. Whitt. Fluid models for multiserver queues with abandonments. *Operations Research*, 54:37–54, 2006.

[37] S. Zeltyn. *Call centers with impatient customers: exact analysis and many-server asymptotics of the* M/M/N+G *queue*. Research thesis, Israel Institute of Technology, October 2004.