Robust Planning for Unmanned Underwater Vehicles

by

ENS Emily Anne Frost, USN

B.S., United States Naval Academy (2011)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author ....
Sloan School of Management
May 10, 2013

Certified by ....
Michael Ricard
Distinguished Member of the Technical Staff, The Charles Stark Draper Laboratory
Technical Supervisor

Certified by ....
Dimitris Bertsimas
Boeing Professor of Operations Research, Sloan School of Management
Co-Director, Operations Research Center
Thesis Supervisor

Certified by ....
Julie Shah
Assistant Professor, Department of Aeronautics and Astronautics
Thesis Supervisor

Accepted by ....
Patrick Jaillet
Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science
Co-Director, Operations Research Center

Robust Planning for Unmanned Underwater Vehicles

by

ENS Emily Anne Frost, USN

B.S., United States Naval Academy (2011)

Submitted to the Sloan School of Management
on May 10, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Operations Research

Abstract

In this thesis, I design and implement a novel method of schedule and path selection between predetermined waypoints for unmanned underwater vehicles under uncertainty. The problem is first formulated as a mixed-integer optimization model and subsequently uncertainty is addressed using a robust optimization approach. Solutions were tested through simulation and computational results are presented which indicate that the robust approach handles larger problems than could previously be solved in a reasonable running time while preserving a high level of robustness.

This thesis demonstrates that the robust methods presented can solve realistic-sized problems in reasonable runtimes – a median of ten minutes and a mean of thirty minutes for 32 tasks – and that the methods perform well both in terms of expected reward and robustness to disturbances in the environment. The latter two results are obtained by simulating solutions given by the deterministic method, a naive robust method, and finally the two restricted affine robust policies. The two restricted affine policies consistently show an expected reward of nearly 100%, while the deterministic and naive robust methods achieve approximately 50% of maximum reward possible.

Name: Michael Ricard
Title: Distinguished Member of the Technical Staff, The Charles Stark Draper Laboratory
Technical Supervisor

Name: Dimitris Bertsimas
Title: Boeing Professor of Operations Research, Sloan School of Management
Co-Director, Operations Research Center
Thesis Supervisor

Name: Julie Shah
Title: Assistant Professor, Department of Aeronautics and Astronautics
Thesis Supervisor

Acknowledgments

As in an Oscar speech, there are many people who helped me along the way. Some notables: Michael Ricard, for going above and beyond the requirements of serving as an advisor, and spending considerable amounts of his personal time troubleshooting my problems; Julie Shah, for providing a warm, welcoming, and above all productive environment at MIT in which to work, and a valuable outside perspective on the technical portions of the thesis; and Dimitris Bertsimas, for guiding the more theoretical aspects of the thesis.

I would also like to thank the Navy and Draper Laboratory for enabling me to conduct this research. Finally, everyone, friends and family, who listened to me grouch about this amazing opportunity and refrained from reminding me at every turn how lucky I am. Rest assured, I know it.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1 Introduction **13**

1.1 Problem Motivation 13

1.2 Literature Review 14

1.3 Contributions 15

1.4 Structure 15

2 Background in Robust Optimization **17**

2.1 Theory of Cardinality-Constrained Robust Optimization 17

2.2 Example of Cardinality-Constrained Robust Optimization 20

3 A Deterministic Model **25**

3.1 Formulation 26

3.1.1 An Example 28

4 Models Under Uncertainty **31**

4.1 Modeling uncertainty 31

4.1.1 Full affine policy 33

4.1.2 All Incoming Arcs affine policy 34

4.1.3 h-Nearest Neighbors affine policy 35

4.2 Robust Counterparts 35

4.2.1 Naive approach 35

4.2.2 Full affine policy 37

4.2.3 Restricted affine policies 42

5 Computational Results **45**

5.1 Runtime results 46

5.2 Solution quality results 50

6 Conclusions **57**

List of Figures

3-1 Example problem with four total tasks. 29

5-1 Median runtime shown graphically for the deterministic, naive, All Incoming Arcs, and h-Nearest Neighbors methods. 48

5-2 Rate of improvement shown for seven randomly selected problems with 22 tasks, with the h-Nearest Neighbors method. Each color indicates a separate problem instance. 49

5-3 Rate of improvement shown for eleven randomly selected problems with 22 tasks, with the deterministic method. Each color indicates a separate problem instance. 49

5-4 This flowchart shows the simulation process. Arcs represent information being passed, blocks represent executable steps. The process takes place once a simulated minute. 50

5-5 Normalized expected reward for each method for each task size, with standard deviation bars. 51

5-6 Number of failures in each decile of the overall solution time for the deterministic method, size 22 tasks. 54

5-7 Number of failures in each decile of the overall solution time for the All Incoming Arcs method, size 17 tasks. 54

5-8 Number of failures in each decile of the overall solution time for the h-Nearest Neighbors method, size 22 tasks. 55

5-9 Median failure points for the deterministic, naive, All Incoming Arcs, and h-Nearest Neighbors methods. 56

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

5.1 Mean and standard deviation, reported in seconds, of runtime of various problem sizes. "All" indicates the All Incoming Arcs method, and "h-NN" indicates the h-Nearest Neighbors method. 46

5.2 First, second, and third quartiles, reported in seconds, of runtime of various problem sizes. "All" indicates the All Incoming Arcs method, and "h-NN" indicates the h-Nearest Neighbors method. 47

5.3 Normalized expected reward (objective function value) for each method for each task size, based on simulation. 51

5.4 First, second, and third quartiles of expected reward normalized to 1. . . . 51

5.5 Mean and standard deviation of number of failures over 50 problems of each size, simulated 200 times each. The average number of failures is out of 200 trials. 52

5.6 First, second, and third quartiles of number of failures over 50 problems of each size, simulated 200 times each. The number of failures is out of 200 trials. 52

5.7 Mean and standard deviation of failure points over 50 problems of each size, simulated 200 times each. 53

5.8 First, second, and third quartiles of failure points over 50 problems of each size, simulated 200 times each. 53

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

Robust planning aims to take into account the uncertainties inherent in a problem's environment in order to deliver a solution which will remain feasible and near-optimal even when adverse circumstances occur. This thesis applies robust planning specifically to the undersea environment, where there are many sources of uncertainty.

1.1 Problem Motivation

Unmanned underwater vehicles (UUVs) are already in use in the US Navy for short-duration missions such as local seafloor mapping and short-term surveillance [18, 15]. The Navy's utilization of these vehicles, however, is restricted by command and control challenges: there is no current technology which allows reliable, secure, and covert communication between entities above the surface of the water and the submerged UUV, or even between multiple submerged UUVs, unlike unmanned aerial vehicles (UAVs). Navy leadership envisions UUVs capable of being sent on 60-90-day missions with minimal human supervision, with similar success rates as manned submarines sent out for similar periods (see [17]). The major hardware obstacle at this time is energy density of battery technologies, but the major software obstacle is reliable path planning that is intelligible to the human tasking the UUV, as stated by the current Chief of Naval Operations in [13] and earlier on in [18].

This question of reliability is the main challenge addressed in this thesis. In order to confidently send UUVs out on 60-90-day unsupervised missions, the mission commander must be certain that the vehicle will return to its stated rendezvous location by the rendezvous time. Missing this rendezvous can imperil manned vessels and their crews by forcing them to remain on station waiting for the UUV, or to cast about in a potentially hostile area searching for the UUV. Therefore it is necessary to have planning software which accounts for these uncertainties and gives the vehicle a schedule and path which ensures its timely arrival at the rendezvous point.

This planning problem is rife with uncertainties inherent to the undersea environment: deep-ocean currents, tidal currents, surface traffic, etc. The planning problem is also complicated by the fact that any UUV in this situation would naturally be over-tasked at the outset of the mission, in the sense of being assigned more tasks than mission planners thought could be completed. This practice ensures that units will experience minimal idle time: some tasks might be impossible to complete for unforeseen reasons, and it is desirable for the UUV to be able to choose a substitute task in this case. Therefore, the planning problem is an optimal task allocation problem, with an objective function that maximizes reward collected for executing individual tasks.

Between the inherent uncertainties and the task-choosing aspect of the problem, deterministic solutions are intuitively unlikely to be successful in terms of arriving at tasks on time according to the computed schedule. Users also would prefer that a vehicle stick to one optimal or near-optimal solution, rather than "jittering" between highly divergent solutions whenever environmental conditions change. Therefore, an approach which simultaneously handles the uncertainty while not sacrificing too much optimality is necessary.

1.2 Literature Review

Route optimization for UUVs under uncertainty has previously been addressed by Cates [9] and Crimmel [10]. In [9], Cates models the problem space as a Time Expanding Network with discrete probabilities of arriving at the next task on time assigned to each enumerated potential travel path of the UUV. This approach provides probabilistic guarantees as to the reliability of the chosen optimal path, but results in low tractability as the number of nodes (tasks) grows: this approach could handle problems of up to 15 tasks in a tractable way. In [10], Crimmel works from the same framework, adding a choice of navigation points for the vehicle. This approach provides greater expressivity for many real world problems, especially in terms of the UUV's navigational requirements, but also results in low tractability, approximately ten tasks, for provably optimal solutions. In [10], Crimmel additionally provides approximate dynamic programming methods for choosing a good task route out of a precomputed group of potential task routes. For a small number of about ten task routes, the approach remains tractable for up to 30 tasks.

This thesis proposes a robust optimization approach to handle the uncertainty in execution times. It demonstrates improved tractability and robustness of the solution over [9] and [10]. Specifically it proposes affinely adaptive policies motivated by the successful application of such policies in [12, 6, 2]. An affine policy is one in which decision variables are formulated as affine combinations of uncertain quantities of the problem. These reformulation techniques can simplify the problem and allow use of robust optimization approaches. In [6], the authors detail the performance limitations of affine policies for a two-stage adaptive linear optimization problem under uncertainty. In [12], the authors investigate the performance and tractability of affine policies for multistage robust optimization problems under uncertainty. Although the authors show that in the worst case the best affine policy is suboptimal, they show also that utilizing affine policies improves tractability in practice without sacrificing too much performance.

1.3 Contributions

The approach presented in this thesis does not use time expanding networks used in [9] and [10]. It features higher tractability in practice, along with modeling flexibility that dynamic programming approaches do not exhibit. In addition, robustness implies that the vehicle will be more likely to reach its rendezvous point on time, if planners have appropriately accounted for uncertainty in the environment. The robust formulation applies polyhedral uncertainty, as discussed in [7] and [11], to affine reformulations of the decision variables.

The thesis also presents restricted affine policies which further improve tractability in practice with minimal performance gaps, shown empirically. These advantages result in an ability to solve larger problems than previously could be solved under uncertainty, in a tractable way in terms of runtime.

1.4 Structure

Chapter 2 provides the theoretical background of the robust optimization methods used in this thesis. Chapter 3 lays out the deterministic mixed-integer formulation of the UUV problem. Chapter 4 discusses the full affine method and two restricted affine methods, which trade optimality for faster computation and work well in practice for problems of realistic size. Chapter 5 presents computational results from the deterministic method, a naive robust method, and finally two affine robust policies. Computational results are presented in terms of computation time and in terms of quality of solution, measured by a discrete event simulator of a UUV mission. Chapter 6 contains conclusions.

Chapter 2

Background in Robust Optimization

In this chapter the basics of robust optimization needed to follow the approaches described in the rest of this thesis are outlined. The thesis applies cardinality-constrained robust optimization, described by Bertsimas and Sim in [7], to four different methods in the course of the thesis. This chapter will lay out the basic theory behind cardinality-constrained robust optimization and also provides a simple example of its use.

2.1 Theory of Cardinality-Constrained Robust Optimization

This approach applies cardinality-constrained robust optimization. This framework encodes uncertainty in a row-wise fashion, with an uncertainty budget (Γ_i) for each row i which allows the user to adjust the level of risk (or risk aversion) desired for the uncertain quantities in that row i. For example, if there are m uncertain coefficients in row i and the user sets $\Gamma_i = m$, the user is indicating that he wishes to protect against the worst case scenario (worst-case optimization). If the user sets $\Gamma_i = 0$, he is indicating that he believes there is no uncertainty in row i; a value of 0 in this framework corresponds to the deterministic case. The subset of uncertain coefficients in row i is called J_i; the further subset of uncertain coefficients which we would like to allow to vary is called S_i, where $S_i \in J_i$ and $|S_i| = \Gamma_i$. Note that the user does not explicitly choose the elements of S_i.

The range of potential uncertainty is described by symmetric box-type uncertainty sets for each uncertain coefficient in the A matrix. For each uncertain coefficient a_{ij}, $\bar{a_{ij}}$ is the nominal value, $\hat{a_{ij}}$ is the maximum divergence, and $\tilde{a_{ij}}$ denotes the actual uncertain value. The uncertainty set, then, says that $\tilde{a_{ij}} \in [\bar{a_{ij}} - \hat{a_{ij}}, \bar{a_{ij}} + \hat{a_{ij}}]$.

Bertsimas and Sim demonstrated a method of handling the robust formulation described in [7] as a single linear optimization problem, which empirically demonstrated significant improvements in computational tractability. The robust formulation is below:

$$\begin{array}{lll} \max & c^T x & \\ \text{subject to} & \sum_j \bar{a_{ij}} x_j + max_{\{S_i \subseteq J_i : |S_i| = \Gamma_i\}} \sum_{j \in S_i} \hat{a_{ij}} y_j & \leq b_i, \ 1 \leq i \leq m \\ & -y_j \leq x_j \leq y_j & 1 \leq j \leq n \\ & l \leq x \leq u, \ y \geq 0 & \end{array}$$

where the decision variables are x_j and y_j.

In words, this formulation says the following: maximize the given objective function, subject to the nominal constraints plus some protective quantity. This protective quantity $max_{\{S_i \subseteq J_i : |S_i| = \Gamma_i\}} \sum_{j \in S_i} \hat{a_{ij}} y_j$ is described by the uncertainty sets ($[\bar{a_{ij}} - \hat{a_{ij}}, \bar{a_{ij}} + \hat{a_{ij}}]$ for each ij) and by the risk budget (Γ_i for each row i).

For each row i, this quantity is described as a protection function $\beta_i(x)$, namely

$$\begin{array}{rll} \beta_i(x) = \max & \sum_{j \in J_i} \hat{a_{ij}} |x_j^*| z_{ij} & \\ \text{subject to} & \sum_{j \in J_i} z_{ij} & \leq \Gamma_i \\ & 0 \leq z_{ij} \leq 1 & \forall j \in J_i \end{array}$$

Essentially, the new decision variable z_{ij} acts as a weight for each uncertain coefficient, in the sense that it describes as a proportion of the total maximum divergence $\hat{a_{ij}}$ how far $\tilde{a_{ij}}$ is from $\bar{a_{ij}}$; hence the constraint that all z_{ij} values must sum to Γ_i, the uncertainty budget defined by the user for this row i.

Now, from Lemma 2.1 and Theorem 2.1 in [16], if there are $x \in \Re^{n \times 1}$ a vector of decision variables; $c \in \Re^{n \times 1}$, $b \in \Re^{m \times 1}$, $G \in \Re^{l \times (m \cdot n)}$, and $d \in \Re^{l \times 1}$ given data; and $\tilde{A} \in \Re^{m \times n}$ a matrix of uncertain coefficients, then the problem

$$\begin{array}{lll} \max & c^T x & \\ \text{subject to} & \tilde{A} x & \leq b \end{array}$$

$$x \in P^x$$
$$\forall \tilde{A} \text{ where } \quad G \cdot vec(\tilde{A}) \leq d$$

is equivalent to

$$\begin{array}{lll} \max & c^T x & \\ \text{subject to} & (p^i)^T G = (x_i)^T, & i = 1, ..., m \\ & (p^i)^T d \leq b_i, & i = 1, ..., m \\ & p^i \geq 0, & i = 1, ..., m \\ & x \in P^x & \end{array}$$

where $p^i \in \Re^{l \times 1}$ and $x_i \in \Re^{(m \cdot n) \times 1}$, $i = 1, ..., m$ are vectors of decision variables. Essentially, for the robust problem format, this theorem implies that strong duality applies: by strong duality, if we have a primal and a dual problem where both are feasible and bounded, the optimal objective function values are the same. The above theorem states that an equivalent problem can be formulated with the original objective function while replacing the constraints with the relevant dual constraints. In turn, this reformulation means the nonlinear aspects of the protection function $\beta_i(x)$ can be transformed into a tractable linear optimization problem as follows.

Since the protection function problem $\beta_i(x)$ is feasible and bounded, take the dual:

$$\begin{array}{lll} \min & \Gamma_i q_i + \sum_{j \in J_i} r_{ij} & \\ \text{subject to} & q_i + r_{ij} & \geq \hat{a_{ij}} |x_j^*| \ \forall j \in J_i \\ & r_{ij} \geq 0 & \forall j \in J_i \\ & q_i \geq 0 & \forall i \end{array}$$

Now since the dual problem is feasible and bounded, the objective function value is equal to the protection function's objective function value (and hence generating the appropriate level of conservatism required by the user's chosen Γ values). Therefore according to [16], the constraints of the dual problem can be substituted for the original subproblem

$(max_{\{S_i \subseteq J_i : |S_i| = \Gamma_i\}} \sum_{j \in S_i} a_{i\hat{j}} y_j)$ in the first robust formulation:

$$\begin{array}{lll} \max & c^T x & \\ \text{subject to} & \sum_j \bar{a_{ij}} x_j + q_i \Gamma_i + \sum_j r_{ij} & \leq b_i, \ \forall i \\ & q_i + r_{ij} & \geq \hat{a_{ij}} y_j, \ \forall i, \ j \\ & -y_j \leq x_j \leq y_j & \forall j \\ & l \leq x \leq u, \ q \geq 0, \ r \geq 0, & y \geq 0 \end{array}$$

This formulation is a single linear optimization problem, with the same objective function and equivalent constraints as the original robust formulation.

2.2 Example of Cardinality-Constrained Robust Optimization

The following simple example problem in two dimensions demonstrates the process of generating the robust formulation. This example problem shows uncertainty in the b vector, which is where the uncertainty lies in the problems described later on in this thesis. Uncertainty in the b vector requires a small amount of special handling, which is shown below.

Consider the problem:

$$\begin{array}{lll} \max & 3x_1 + 2x_2 & \\ \text{subject to} & x_1 + 2x_2 & \leq b_1 \\ & 3x_1 - x_2 & \leq b_2 \\ & x_1 \geq 0, & x_2 \geq 0 \\ & & b_1 \in [6, \ 7] \\ & & b_2 \in [9, \ 12] \end{array}$$

First transform the uncertain ranges of the b vector into the symmetric uncertainty sets required by the cardinality-constrained optimization framework:

$$b_1 = 6.5, \quad \hat{b_1} = .5, \quad \tilde{b_1} \in \left[b_1 - \hat{b_1}, \ b_1 + \hat{b_1} \right]$$

$$b_2 = 10.5, \quad \hat{b_2} = 1.5, \quad \tilde{b_2} \in \left[b_2 - \hat{b_2},\, b_2 + \hat{b_2}\right]$$

Next, move the b vector into the A matrix, since cardinality-constrained optimization is designed to handle uncertainty in the A matrix rather than in the b vector. Manage this by adding one column to the A matrix: the uncertain b values are the coefficients of this column, and the extra variable (x_3) is constrained to be 1 at all times.

$$\begin{array}{lll}
\max & 3x_1 + 2x_2 & \\
\text{subject to} & x_1 + 2x_2 - \tilde{b_1}x_3 & \leq 0 \\
& 3x_1 - x_2 - \tilde{b_2}x_3 & \leq 0 \\
& x_1,\, x_2 \geq 0 & x_3 = 1 \\
& \tilde{b_1} \in \left[b_1 - \hat{b_1},\, b_1 + \hat{b_1}\right] & \\
& \tilde{b_2} \in \left[b_2 - \hat{b_2},\, b_2 + \hat{b_2}\right] &
\end{array}$$

Now recast the above nonlinear problem in the notation of cardinality-constrained optimization:

$$\begin{array}{lll}
\max & 3x_1 + 2x_2 & \\
\text{subject to} & x_1 + 2x_2 - b_1x_3 + max_{\{S_1 \subseteq J_1 : |S_1| = \Gamma_1\}} \sum_{j \in S_1} \hat{b_1} y_3 & \leq 0 \\
& 3x_1 - x_2 - b_2x_3 + max_{\{S_2 \subseteq J_2 : |S_2| = \Gamma_2\}} \sum_{j \in S_2} \hat{b_2} y_3 & \leq 0 \\
& -y_3 \leq x_3 \leq y_3,\; x_3 = 1 & x_1,\, x_2 \geq 0
\end{array}$$

Here the two protection functions $\beta_1(x)$ and $\beta_2(x)$ are apparent, and can be written as linear optimization problems. The example uses w_i for the weight variables (the decision variables in the protection function linear programming problems) because z_i is used elsewhere in the primary research problem notation.

$\beta_1(x^*, \Gamma_1)$:

$$\begin{array}{lll}
\max & \hat{b_1}|x_3|w_1 & \\
\text{subject to} & w_1 & \leq \Gamma_1 \\
& 0 \leq w_1 \leq 1 &
\end{array}$$

$\beta_2(x^*, \Gamma_2)$:

$$\begin{array}{rcl} \max & \hat{b_2}|x_3|w_2 & \\ \text{subject to} & w_2 & \leq \Gamma_2 \\ & 0 \leq w_2 \leq 1 & \end{array}$$

Then take the dual of each problem:

$D\beta_1(x^*, \Gamma_1)$:

$$\begin{array}{rcl} \min & \Gamma_1 q_1 + r_{13} & \\ \text{subject to} & q_1 + r_{13} & \geq \hat{b_1}|x_3| \\ & q_1 \geq 0, & r_{13} \geq 0 \end{array}$$

$D\beta_2(x^*, \Gamma_2)$:

$$\begin{array}{rcl} \min & \Gamma_2 q_2 + r_{23} & \\ \text{subject to} & q_2 + r_{23} & \geq \hat{b_2}|x_3| \\ & q_2 \geq 0, & r_{23} \geq 0 \end{array}$$

where there is only one r variable per row because there is only one uncertain coefficient per row of the A matrix. In any case, there would be only one q variable per row, but there are as many r variables in a given row i as uncertain coefficients in that row of the A matrix.

So after substituting the dual constraint conditions back into the original robust formulation, obtain the following:

$$\begin{array}{rcl} \max & 3x_1 + 2x_1 & \\ \text{subject to} & x_1 + 2x_2 - 6.5x_3 + (\Gamma_1 q_1 + r_{13}) & \leq 0 \\ & 3x_1 - x_2 - 10.5x_3 + (\Gamma_2 q_2 + r_{23}) & \leq 0 \\ & q_1 + r_{13} & \geq .5y_3 \\ & q_2 + r_{23} & \geq 1.5y_3 \\ & x_1,\, x_2 \geq 0,\, x_3 = 1 & q_1,\, q_2 \geq 0 \end{array}$$

$$r_{13}, r_{23} \geq 0 \qquad\qquad -y_3 \leq x_3 \leq y_3$$

recalling that $b_1 = 6.5$, $\hat{b_1} = .5$, $b_2 = 10.5$, and $\hat{b_2} = 1.5$.

This method of obtaining robust counterparts will be used throughout Chapter 4 to obtain robust formulations for each of four approaches to handling the uncertainty in the underwater environment.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

A Deterministic Model

This chapter describes the problem and presents an initial formulation of the deterministic model of the unmanned underwater vehicle (UUV) planning problem under uncertainty. In succinct terms, the problem is to develop a schedule for a UUV which maximizes reward collected, by completing tasks in an optimal manner, and which reaches the rendezvous point on time. This final rendezvous constraint is a hard constraint, as described in section 1.1.

The problem is first formulated as a mixed-integer optimization problem. This initial approach moved away from the fundamentally probabilistic approach inherent in previous approaches [9, 10]. The use of uncertainty sets to address randomness in the real-world environment permits the use of robust optimization to generate robust solutions in a tractable way.

The deterministic model is developed for a model with n tasks, including a start and end task which are potentially artificially imposed. There may be time-related constraints on these tasks, such as windows within which certain tasks must be completed if they are to be done at all. These windows might be absolute, or relative to completion of another task. These potential constraints are included in the model.

This chapter details the relevant quantities in the deterministic formulation along with its structure, and also gives an example to illustrate the function of the quantities described.

3.1 Formulation

This formulation assumes that there are n tasks, including an explicit start task and end task. It works from a basic graph formulation, with nodes i and arcs (i,j), where arc (i,j) represents physical travel from node i to node j. The formulation also assumes that arriving at node i is synonymous with beginning task i. When there are n tasks, task 0 is the origin point of the plan and task n is defined as arriving at the final rendezvous location.
The data are:

- c_{ij}, a value indicating the sum of the duration of task i and the travel time of the UUV from node i to node j.

- (a_i, b_i), a temporal interval constraint specifying the allowable timeframe for executing task i (alternatively, arriving at node i). There may not exist a (a_i, b_i) pair for every task. This pair indicates that the UUV must begin task i no earlier than time a_i after the start of the plan and no later than time b_i after the start of the plan if task i is executed, namely, $a_i \leq b_i$. The set B_i indicates the set of nodes for which a (a_i, b_i) pair exists.

- (l_{ij}, u_{ij}), a temporal interval constraint specifying the allowable timeframe for executing task j relative to the completion time of task i. There may not exist a (l_{ij}, u_{ij}) pair for every arc ij. This pair indicates that the UUV must begin task j no earlier than l_{ij} time units after task i, and no later than u_{ij} time units. after task i if both tasks i and j are completed, namely, $l_{ij} \leq u_{ij}$. L_{ij} represents the set of arcs ij for which a (l_{ij}, u_{ij}) pair exists.

- f_i, a reward for completing each task. The objective function is to maximize the sum of these rewards.

- T, an upper bound in mission elapsed time, or the time elapsed since executing the start task.

The decision variables are:

- z_i, a binary variable, where a value of 1 indicates that node (task) i is visited (that the UUV completes task i), and a value of 0 indicates that node (task) i is not visited, for all $i \in \{1, ..., n\}$.

- y_{ij}, a binary variable, where a value of 1 indicates that arc (i, j) is selected for travel, and a value of 0 indicates that arc ij is not selected, for all $i \in \{0, ..., n-1\}$, $j \in \{1, ..., n\}$. By construction, there are no incoming arcs to the start node, node 0, and no outgoing arcs from the end node, node n.

- p_i, a continuous variable indicating the time at which the UUV begins task i, for all $i \in \{1, ..., n\}$.

The objective function is $\sum f_i z_i$ which will be maximized.

The deterministic model, in compact form utilizing big-M constraints where there are n tasks, follows. M is a number large enough to enforce binary constraints; in this case, a number slightly larger than T is used.

$$\begin{array}{llll}
\text{maximize} & \displaystyle\sum_{i=1}^{n} f_i z_i & & (3.1) \\
\text{subject to} & y_{ii} = 0 & \forall i \in \{1, ..., n-1\} & (3.2) \\
& \displaystyle\sum_{j=1}^{n} y_{ij} = z_i & \forall i \in \{0, ..., n-1\} & (3.3) \\
& \displaystyle\sum_{i=0}^{n-1} y_{ij} = z_j & \forall j \in \{1, ..., n\} & (3.4) \\
& p_j - p_i \geq c_{ij} - M(1 - y_{ij}) & \forall i \in \{0, ..., n-1\}, j \in \{1, ..., n\} & (3.5) \\
& p_j - p_i \geq l_{ij} - M(2 - z_i - z_j) & \forall i \in L_{ij} & (3.6) \\
& p_j - p_i \leq u_{ij} + M(2 - z_i - z_j) & \forall j \in L_{ij} & (3.7) \\
& p_i - p_0 \geq a_i - M(1 - z_i) & \forall i \in B_i & (3.8) \\
& p_i - p_0 \leq b_i + M(1 - z_i) & \forall i \in B_i & (3.9) \\
& p_n - p_0 \leq T & & (3.10) \\
& z_0, z_n = 1 & z_i \in \{0, 1\} \forall i \neq 0, n & (3.11) \\
& y_{ij} \in \{0, 1\} & \forall i \in \{0, ..., n-1\}, j \in \{1, ..., n\} & (3.12) \\
& p_i \geq 0, & p_i \leq T & (3.13)
\end{array}$$

The reward functions are set by the user and can be as simple as a constant value assigned to each node, which is collected when that node is visited.

Constraint (3.2) forbids the UUV to travel from task i back to task i; (3.3) and (3.4) dictate exactly one incoming arc and one outgoing arc for tasks visited, and conversely exactly zero incoming and outgoing arcs for tasks not visited. They are adjusted to account for the starting task having one outgoing and zero incoming arcs, and for the ending task having one incoming and zero outgoing arcs. These constraints ensure that the path chosen by the model is a coherent path with no disjoint cycles. They also ensure that the UUV does not repeat tasks.

Constraints (3.5)-(3.10) and constraint (3.13) ensure that the schedule obeys the chronological limitations given by the data. For example, in (3.5), if the arc from i to j is chosen, then the timestamp of task j should be at least equal to the timestamp of task i, plus the sum of the travel time from task i to task j and the completion time of task i (in this model we fold task duration and travel time into the single quantity c_{ij}). Constraint (3.11) forces completion of the start and end tasks. The subsequent constraints encode this consideration for other chronological constraints, including inter-task constraints and the final elapsed-time deadline T.

3.1.1 An Example

To better understand the formulation (3.1)-(3.13), consider a UUV with four potential tasks, including a starting point and a final rendezvous point. In Figure 3-1, circles represent tasks, and arrows represent possible travel arcs. Consider the following requirements:

- The vehicle must arrive at Task 3 (the end task) within 12 time units from the mission start.
- Task 1 takes 2 time units to complete.
- Task 2 takes 1 time unit to complete.
- Task 1 must be completed between 2 and 5 time units from the mission start if Task 1 is completed: $2 \leq p_1 \leq 5$.
- Task 2 must be completed within 4 time units after Task 1 if both Task 1 and Task 2 are completed, and Task 2 must be completed after Task 1 if both Task 2 and Task 1 are completed: $0 \leq p_2 - p_1 \leq 4$.
- There is a reward of 3 for completing Task 1.

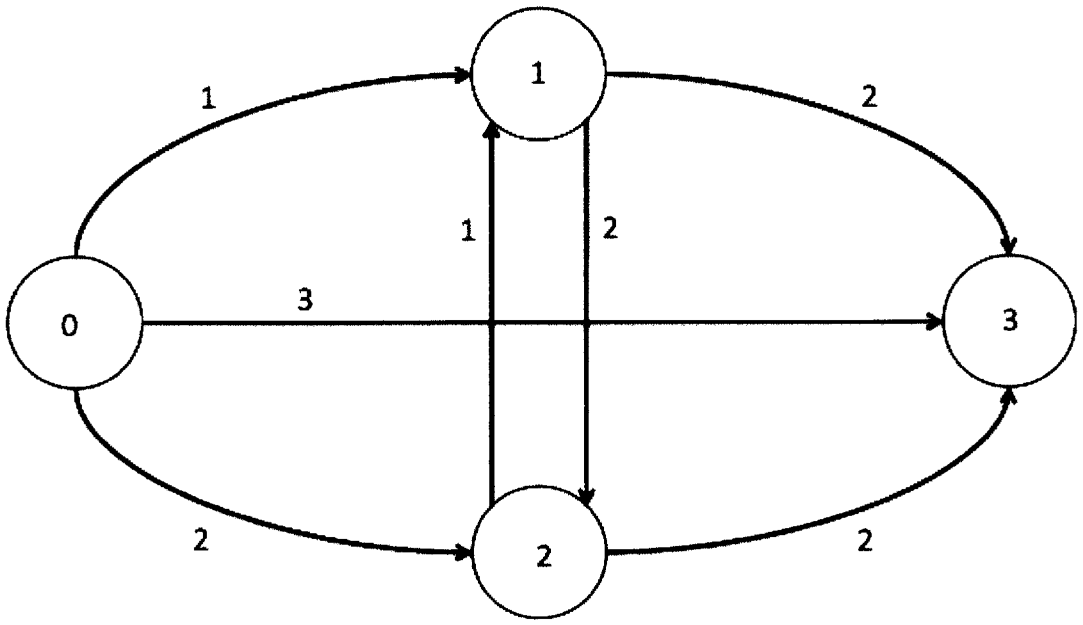

Figure 3-1: Example problem with four total tasks.

- There is a reward of 5 for completing Task 2.

In Figure 3-1, the numbers along travel arcs represent the number of time units it takes to travel along that arc.

In the deterministic model, in order to minimize the number of nodes and thus the problem size, the c_{ij} quantity is calculated for each node-arc pair. For example, $c_{12} = 2+1 = 3$, because Task 1 takes two time units to complete and the travel arc y_{12} takes one time unit to traverse. Using these quantities, the following formulation corresponds to the example problem shown.

$$\begin{aligned} \text{maximize} \quad & 0z_0 + 3z_1 + 5z_2 + 0z_3 \\ \text{subject to} \quad & y_{11} = 0,\ y_{22} = 0 \end{aligned}$$

$$\begin{aligned}
y_{01} + y_{02} + y_{03} &= z_0 \\
y_{12} + y_{13} &= z_1 \\
y_{21} + y_{23} &= z_2 \\
y_{01} + y_{21} &= z_1 \\
y_{02} + y_{12} &= z_2 \\
y_{03} + y_{13} + y_{23} &= z_3 \\
p_1 - p_0 &\geq 3 - M\left(1 - y_{01}\right) \\
p_2 - p_0 &\geq 2 - M\left(1 - y_{02}\right) \\
p_3 - p_0 &\geq 3 - M\left(1 - y_{03}\right) \\
p_2 - p_1 &\geq 3 - M\left(1 - y_{12}\right) \\
p_1 - p_2 &\geq 3 - M\left(1 - y_{21}\right) \\
p_3 - p_1 &\geq 4 - M\left(1 - y_{13}\right) \\
p_3 - p_2 &\geq 3 - M\left(1 - y_{23}\right) \\
p_2 - p_1 &\geq 0 - M\left(2 - z_1 - z_2\right) \\
p_2 - p_1 &\leq 4 - M\left(2 - z_1 - z_2\right) \\
p_1 - p_0 &\geq 2 - M\left(1 - z_1\right) \\
p_1 - p_0 &\leq 5 + M\left(1 - z_1\right) \\
p_3 - p_0 &\geq 0 \\
p_3 - p_0 &\leq 12 \\
z_0,\ z_3 = 1 & \quad z_1,\ z_2 \in \{0,1\} \\
y_{ij} \in \{0,1\} & \quad \forall i \in \{0,1,2\}, \forall j \in \{1,2,3\} \\
p_i \geq 0, & \quad \forall i \in \{0,1,2\} \\
p_i \leq T & \quad \forall i \in \{0,1,2\}
\end{aligned}$$

The next chapter presents approaches for handling uncertainty in task completion times and travel times between tasks.

Chapter 4

Models Under Uncertainty

This thesis models a problem where all uncertainty comes from the environment (tides, wind currents, etc.) These sources of uncertainty only influence task duration and travel times between tasks. This model assumes that data such as B_i and L_{ij}, temporal constraints between tasks or delineating exact execution windows for particular tasks, do not have uncertainty. Therefore, there are two sources of uncertainty in this problem: uncertainty in travel times between tasks, and uncertainty in task completion times. By this assumption, and because the model folds these two pieces of data into the quantity c_{ij}, the result is only one uncertain value per row. One important implication of these modeling choices is that only the constraint relating timestamps to travel times will incur uncertainty when the formulation shifts to the robust model (constraint (3.5) in Chapter 3).

For this example, consider the following uncertainties:

- Task 1 may take from 1 to 3 time units to complete.
- Task 2 may take from 1.75 to 2.25 time units to complete.
- It may take from 1 to 3 time units to travel from Task 1 to Task 3.
- It may take from 1.5 to 2.5 time units to travel from Task 2 to Task 3.

4.1 Modeling uncertainty

This model uses uncertainty sets rather than probabilistic frameworks to model the uncertainty in the problem. Due to the sources of uncertainty in the problem, it is virtually impossible to appropriately model the resulting randomness probabilistically. Namely, the vehicle may encounter disturbances in task completion and travel times due to such diverse sources as tides, wind-driven currents, deep ocean currents, and ocean-going traffic, among others. Furthermore, some of these sources have geography-based and time-based dependencies, additionally complicating the modeling process. Therefore, this thesis uses uncertainty sets constructed using a set of nominal travel times $\bar{c}_{ij}$ and a set of maximum possible divergences from those nominal values $\hat{c}_{ij}$ expected given reasonable knowledge of the intended travel environment. Then each uncertain value $\tilde{c}_{ij}$ belongs to an interval $U_{ij} = [\bar{c}_{ij} - \hat{c}_{ij}, \bar{c}_{ij} + \hat{c}_{ij}]$ which is the uncertainty set for that particular uncertain coefficient.

For this application, this type of "box" uncertainty set is appropriate because pelagic travel times are often affected by factors which take the form of an average quantity plus or minus some disturbance. Tidal currents are a good example: a typical tidal cycle may have an average current speed of 0 knots, plus or minus 2 knots depending on the state of the cycle.

A Γ value is chosen for each row of the A matrix containing an uncertain coefficient. In my modeling framework, the Γ value represents the number of uncertain coefficients which are allowed to take their worst-case value. Therefore it must be less than or equal to the number of uncertain coefficients to the vector of decision variables in each row of the A matrix.

As touched on briefly above, this model uses cardinality-constrained robust optimization to handle the uncertainty in the problem. However, when there is only one uncertain coefficient per row, cardinality-constrained optimization is problematic. The Γ value is meant to spread risk over multiple uncertain coefficients per row; if there is only one uncertain coefficient, then the user is essentially just setting a new, more conservative $\bar{c}_{ij}$ (nominal value) for the travel time in that row.

Consider this example. For the path from Task 1 to Task 2, the given uncertainty implies that $c_{12} \in [2, 4]$. For symmetrical box uncertainty sets, it follows that $\hat{c}_{12} = 1$ and $\bar{c}_{12} = 3$. If the user chooses $\Gamma_{12} = 0.5$, then the effective value of c_{12} will be $3 + (0.5 * 1) = 3.5$. In other words, the Γ value for one uncertain coefficient does not add robustness in the sense of spreading risk; it just makes the solution value more conservative.

The computational results from the naive approach demonstrate these weaknesses, detailed later in this chapter.

The uncertainty is modeled in four different ways: a naive approach, which does not reformulate any decision variables; a full affine policy, which takes every possible travel arc into account; an All Incoming Arcs affine policy, which takes into account all incoming arcs to a given node; and an h-Nearest Neighbors policy, which takes into account incoming arcs from the h nearest nodes to a given node. The next sections describe the uncertainty sets for the three affine policies, which reformulate the decision variable p_i. The uncertainty set for each $\tilde{c}_{ij}$ for the naive approach remains the box uncertainty set discussed above.

4.1.1 Full affine policy

The full affine policy reformulates each p_i as an affine combination of all possible travel times, so that each individual Γ value (one per row) spreads uncertainty over many uncertain coefficients rather than over a single uncertain coefficient. This approach eliminates the effects of having only one uncertain coefficients.

Each timestamp p_k, $\forall k \in (1, ..., n)$ is reformulated in the following manner:

$$p_k = p_0^k + \alpha_{01}^k \tilde{c}_{01} + \alpha_{02}^k \tilde{c}_{02} + \cdots + \alpha_{(n-1)(n)}^k \tilde{c}_{(n-1)(n)}$$

where p_0^k is a decision variable unique to each p_k and each α_{ij}^k is a decision variable, unique to each affine combination. The index (k, i, j) refers to the timestamp k and the arc (i, j). There will be one α_{ij}^k for each timestamp and for each travel arc. Not only does this method spread risk over many uncertain coefficients, rather than having each Γ value correspond directly to one uncertain coefficient, it also retains full flexibility of potential task orderings and selection in the model. Because most tasks are not pre-ordered, any assumptions regarding which travel times are "likely" to contribute to each timestamp restricts the scope of the uncertainty. In order to account for all possible task orderings, each possible travel time is allowed to potentially contribute to each timestamp.

The relevant uncertain constraint follows:

$$p_k - p_l \geq \tilde{c}_{ij} - M(1 - y_{ij})$$

The method retains the box uncertainty sets discussed in section 4.1 for each $\tilde{c}_{ij}$ value, so that the uncertainty set U_1 for the full affine policy and the compact reformulation of each timestamp p_k follows:

$$\begin{aligned} p_k = &\quad \sum_{i,j} \alpha_{ij}^k \tilde{c}_{ij} + p_0^k, \\ \tilde{\mathbf{c}} \in U_1 = &\quad \Big\{ \tilde{c}_{ij} \Big| \Big| \tfrac{\tilde{c}_{ij} - \bar{c}_{ij}}{\hat{c}_{ij}} \Big| \leq 1, \\ &\quad \sum_{i,j} \Big| \frac{\tilde{c}_{ij} - \bar{c}_{ij}}{\hat{c}_{ij}} \Big| \leq \min\left(\Gamma_{kl}, \Gamma_{lk}\right) \Big\}. \end{aligned}$$

4.1.2 All Incoming Arcs affine policy

For this policy, instead of each timestamp being reformulated as an affine combination of all possible travel arcs, each timestamp is reformulated as an affine combination of all travel arcs flowing into that timestamp's corresponding node. In other words, this approach assumes that the uncertainty in the travel arc traversed immediately previous to executing task k has the most influence over the execution time of task k. This approach helps mitigate the problem of large numbers of additional variables, as well as greatly decreasing the number of nonzero coefficients in the A matrix. It does not capture the full complexity of the real-world problem, but the assumption is reasonable for this application.

Specifically, formulate each timestamp as follows:

$$p_k = p_0^k + \alpha_{0k}\tilde{c}_{0k} + \alpha_{1k}\tilde{c}_{0k} + \cdots + \alpha_{(n-1)k}\tilde{c}_{(n-1)k},$$

where, as in section 4.1.1, each p_0^k is a decision variable, and each α_{ik} is a decision variable associated with each travel arc (i, k), where $i = 0, ..., n-1$, $i \neq k$.

Then the uncertainty set U_2 and the compact reformulation of each timestamp p_k for the All Incoming Arcs policy follows:

$$\begin{aligned} p_k = &\quad \sum_{i} \alpha_{ik} \tilde{c}_{ij} + p_0^k \\ \tilde{\mathbf{c}} \in U_2 = &\quad \Big\{ \tilde{c}_{ij} \Big| \Big| \tfrac{\tilde{c}_{ij} - \bar{c}_{ij}}{\hat{c}_{ij}} \Big| \leq 1, \\ &\quad \sum_{i} \Big| \frac{\tilde{c}_{ik} - \bar{c}_{ik}}{\hat{c}_{ik}} \Big| + \sum_{i} \Big| \frac{\tilde{c}_{il} - \bar{c}_{il}}{\hat{c}_{il}} \Big| \leq \Gamma_{kl} \Big\}. \end{aligned}$$

4.1.3 h-Nearest Neighbors affine policy

This policy restricts the arcs which contribute to the affine combination for each timestamp, to include only the incoming arcs from the h nearest nodes to each node. In the case where the end node is one of the h nearest neighbors, the timestamp is formulated as an affine combination of the other $h-1$ nearest neighbors. "Nearest" refers to closest in terms of physical location. H_k is the set of the h nodes nearest to node k.

Then the uncertainty set U_3 and the compact reformulation of each timestamp p_k for the h-Nearest Neighbors policy follows:

$$\begin{aligned} p_k = \quad & \sum_{i \in H_k} \alpha_{ik} \tilde{c}_{ij} + p_0^k, \\ \tilde{\mathbf{c}} \in U_3 = \quad & \Big\{ \tilde{c}_{ij} \Big| \Big| \tfrac{\tilde{c}_{ij} - \bar{c}_{ij}}{\hat{c}_{ij}} \Big| \leq 1, \\ & \sum_{i \in H_k} \left| \frac{\tilde{c}_{ik} - \bar{c}_{ik}}{\hat{c}_{ik}} \right| + \sum_{i \in H_l} \left| \frac{\tilde{c}_{il} - \bar{c}_{il}}{\hat{c}_{il}} \right| \leq \Gamma_{kl} \Big\}. \end{aligned}$$

4.2 Robust Counterparts

This section presents the explicit robust counterparts to the uncertainty sets discussed in Chapter 2.

4.2.1 Naive approach

This initial approach applies cardinality-constrained robust optimization to the deterministic problem, as laid out in Chapter 2. The user chooses a Γ value for each row containing an uncertain coefficient that represents the budget of risk for that row. In this case, that Γ value exactly corresponds to the budget of risk for the one uncertain coefficient in its row; in other words, each Γ_{ij} describes exactly the budget of risk for its corresponding $\tilde{c}_{ij}$ value.

As mentioned above, the $\tilde{c}_{ij}$ values are subtracted over into the A matrix in order to facilitate applying cardinality-constrained robust optimization. Then the relevant constraints with uncertain coefficients (those constraints relating the travel times between tasks) look as follows:

$$p_j - p_i - \tilde{c}_{ij} \geq -M(1 - y_{ij}).$$

The constraint is then multiplied across by -1 in order to reverse the direction of the inequality to match the theoretical basis provided in Chapter 2:

$$p_i - p_j + \bar{c}_{ij} \leq M(1 - y_{ij}).$$

Next a protection function is constructed for each arc ij, namely, β_{ij} :

β_{kl} :

$$\begin{array}{rcl} \max & \hat{c}_{ij}\,|p_{n+1}|\,z_{ij} & \\ \text{subject to} & z_{ij} & \leq \Gamma_{ij} \\ & 0 \leq z_{ij} \leq 1. & \end{array}$$

And take the dual:

$$\begin{array}{rcl} \min & \Gamma_{ij}q_{ij} + r_{ij} & \\ \text{subject to} & q_{ij} + r_{ij} & \geq \hat{c}_{ij} \\ & r_{ij} & \geq 0 \\ & q_{ij} & \geq 0. \end{array}$$

As described above, by strong duality, substitute these constraints back into the original nonlinear uncertain constraint to yield:

$$\begin{array}{cl} p_i - p_j + \bar{c}_{ij} + \Gamma_{ij}q_{ij} + r_{ij} & \leq M(1 - y_{ij}) \\ q_{ij} + r_{ij} & \geq \hat{c}_{ij} \\ r_{ij} & \geq 0 \\ q_{ij} & \geq 0. \end{array}$$

By plugging these new constraints into the deterministic formulation in place of the deterministic travel time constraints, a linear robust formulation is developed:

$$\begin{array}{lll}
\text{maximize} & \sum_{i=1}^{n} f_i z_i & \\
\text{subject to} & y_{ii} = 0 & \forall i \epsilon \{1, ..., n-1\} \\
& \sum_{j=1}^{n} y_{ij} \;= z_i & \forall i \epsilon \{0, ..., n-1\} \\
& \sum_{i=0}^{n-1} y_{ij} \;= z_j & \forall j \epsilon \{1, ..., n\} \\
& p_i - p_j + \bar{c}_{ij} + \Gamma_{ij} q_{ij} + r_{ij} \leq M(1 - y_{ij}) & \forall i \epsilon \{0, ..., n-1\},\, j \epsilon \{1, ..., n\} \\
& q_{ij} + r_{ij} \geq \hat{c}_{ij} & \forall i \epsilon \{0, ..., n-1\},\, j \epsilon \{1, ..., n\} \\
& p_j - p_i \geq l_{ij} - M(2 - z_i - z_j) & \forall i \epsilon L_{ij} \\
& p_j - p_i \leq u_{ij} + M(2 - z_i - z_j) & \forall j \epsilon L_{ij} \\
& p_i - p_0 \geq a_i - M(1 - z_i) & \forall i \epsilon B_i \\
& p_i - p_0 \leq b_i + M(1 - z_i) & \forall i \epsilon B_i \\
& p_n - p_0 & \geq 0 \\
& p_n - p_0 & \leq T \\
& z_0,\, z_n = 1 & z_i \epsilon \{0, 1\}\, \forall i \neq 0, n \\
& y_{ij} \epsilon \{0, 1\} & \forall i \epsilon \{0, ..., n-1\},\, j \epsilon \{1, ..., n\} \\
& r_{ij} \geq 0 & \forall i \epsilon \{0, ..., n-1\},\, j \epsilon \{1, ..., n\} \\
& q_{ij} \geq 0 & \forall i \epsilon \{0, ..., n-1\},\, j \epsilon \{1, ..., n\}.
\end{array}$$

4.2.2 Full affine policy

For the determination of the robust counterparts for the three affine policies, this chapter shows the process once for the most complex case. The process is the same as that outlined in Chapter 2 and section 4.2.1.

Following the convention from section 4.1.1, each uncertain constraint then takes the following form:

$$\begin{aligned}&\left(p_0^k + \alpha_{01}^k \tilde{c}_{01} + \alpha_{02}^k \tilde{c}_{02} + \cdots + \alpha_{(n-1)(n)}^k \tilde{c}_{(n-1)(n)}\right)\\ &\qquad - \left(p_0^l + \alpha_{01}^l \tilde{c}_{01} + \alpha_{02}^l \tilde{c}_{02} + \cdots + \alpha_{(n-1)(n)}^l \tilde{c}_{(n-1)(n)}\right) + \tilde{c}_{kl}\\ &\qquad\qquad \leq M\left(1 - y_{kl}\right) \qquad (4.1)\end{aligned}$$

for all arcs (k,l) and all arcs (i,j), where $k \neq l$ and $i \neq j$.

The robust counterpart based on Equation 4.1, including uncertainty in the $\tilde{c}_{ij}$ quantities, is below:

Theorem 1. *The robust counterpart to Equation 4.1 is*

$$\begin{aligned} p_0^k - p_0^l + \quad & \sum_i \sum_j \alpha_{ij}^k \bar{c}_{ij} - \left(\sum_i \sum_j \alpha_{ij}^l \bar{c}_{ij}\right) + \bar{c}_{kl} & \\ & + \Gamma_{kl} q_{kl} + \sum_i \sum_j r_{kij} + \sum_i \sum_j r_{lij} + r_{kl} \quad \leq M\left(1 - y_{kl}\right) & \\ & q_{kl} + \sum_i \sum_j r_{kij} \geq \hat{c}_{ij} t_{kij} & \forall i,j,k,l, i \neq j \\ & -q_{kl} + \sum_i \sum_j r_{lij} \geq -\hat{c}_{ij} t_{lij} & \forall i,j,k,l, i \neq j \\ & q_{kl} + r_{kl} \geq \hat{c}_{kl} & \\ & -t_{kij} \leq \alpha_{kij}^* \leq t_{kij} & \forall k,l,i,j \\ & q_{kl} \geq 0,\ r_{kij} \geq 0,\ r_{lij} \geq 0,\ t_{kij} \geq 0,\ r_{kl} \geq 0 & \forall k,l,i,j \end{aligned}$$

Proof. Cardinality-constrained optimization as in [7] is applied to these uncertain constraints:

$$\begin{aligned} p_0^k - p_0^l + \sum_i \sum_j \alpha_{ij}^k \bar{c_{ij}} - \left(\sum_i \sum_j \alpha_{ij}^l \bar{c_{ij}}\right) + \bar{c_{kl}} & \\ + \max\left(\sum_i \sum_j \hat{c_{ij}} \left|\alpha_{ij}^k\right| w_{kij} - \sum_i \sum_j \hat{c_{ij}} \left|\alpha_{ij}^l\right| w_{lij} + \hat{c_{kl}} w_{kl}\right) & \\ \leq M\left(1 - y_{kl}\right) & \end{aligned}$$

The protection function, an optimization problem which describes the uncertain aspects of each row of the A matrix as restricted by the risk allocation parameter Γ_{kl}, for each arc kl

where $k \neq l$, for all $k = 0, ..., n-1$, $l = 1, ..., n$, then is:

β_{kl} :

$$\begin{array}{lll} \max & \sum_i \sum_j \hat{c}_{ij} \left|\alpha_{ij}^{k*}\right| z_{kij} - \sum_i \sum_j \hat{c}_{ij} \left|\alpha_{ij}^{l*}\right| z_{lij} + \hat{c}_{kl} z_{kl} & \\ \text{subject to} & \sum_i \sum_j z_{kij} - \sum_i \sum_j z_{lij} + z_{kl} \leq \Gamma_{kl} & 0 \leq z_{kij} \leq 1 \\ & 0 \leq z_{lij} \leq 1 & \\ & 0 \leq z_{kl} \leq 1 & \\ & \forall i = 0, ..., n-1, j = 1, ..., n,\ j \neq i & \end{array}$$

Again take the dual:

$$\begin{array}{lll} \min & \Gamma_{kl} q_{kl} + \sum_i \sum_j r_{kij} + \sum_i \sum_j r_{lij} + r_{kl} & \\ \text{subject to} & q_{kl} + \sum_i \sum_j r_{kij} \geq \hat{c}_{ij} \left|\alpha_{ij}^{k*}\right| & \forall i = 0, ..., n-1, j = 1, ..., n,\ j \neq i \\ & -q_{kl} + \sum_i \sum_j r_{lij} \geq -\hat{c}_{ij} \left|\alpha_{ij}^{l*}\right| & \forall i = 0, ..., n-1, j = 1, ..., n,\ j \neq i \\ & q_{kl} + r_{kl} \geq \hat{c}_{kl} & \\ & -t_{kij} \leq \alpha_{ij}^{k*} \leq t_{kij} & \forall k,\ i = 0, ..., n-1, j = 1, ..., n,\ j \neq i \\ & q_{kl} \geq 0,\ r_{kij} \geq 0,\ r_{lij} \geq 0, r_{kl} \geq 0\, t_{kij} \geq 0 & \forall k, l, i, j. \end{array}$$

Finally, substitute back into the original formulation to give the full robust formulation. The following index sets are assumed: $\forall i \in \{0, ..., n-1\}$, $j \in \{1, ..., n\}$, $\forall k = 0, ..., n-1, l = 1, ..., n$. These sets correspond to the constructed arcs (i, j) of the problem and the available tasks k.

$$\begin{array}{lll} \text{maximize} & \sum_{i=1}^{n} f_i z_i & \\ \text{subject to} & y_{ii} = 0 & \forall i \in \{1, ..., n-1\} \\ & \sum_{j=1}^{n} y_{ij} = z_i & \forall i \in \{0, ..., n-1\} \\ & \sum_{i=0}^{n-1} y_{ij} = z_j & \forall j \in \{1, ..., n\} \end{array}$$

$$
\begin{array}{rll}
p_0^k - p_0^l + & \sum_i \sum_j \alpha_{ij}^k \bar{c}_{ij} - \left(\sum_i \sum_j \alpha_{ij}^l \bar{c}_{ij}\right) + \bar{c}_{kl} & \\
& +\Gamma_{kl} q_{kl} + \sum_i \sum_j r_{kij} + \sum_i \sum_j r_{lij} + r_{kl} & \leq M\left(1 - y_{kl}\right) \\
& q_{kl} + \sum_i \sum_j r_{kij} \geq \hat{c}_{ij} t_{kij} & \forall i, j, k, l, i \neq j \\
& -q_{kl} + \sum_i \sum_j r_{lij} \geq -\hat{c}_{ij} t_{lij} & \forall i, j, k, l, i \neq j \\
& q_{kl} + r_{kl} \geq \hat{c}_{kl} & \\
& -t_{kij} \leq \alpha_{ij}^{k*} \leq t_{kij} & \forall k, l, i, j \\
& p_j - p_i \geq l_{ij} - M(2 - z_i - z_j) & \forall i, j \\
& p_j - p_i \leq u_{ij} + M(2 - z_i - z_j) & \forall i, j \\
& p_i \geq a_i - M(1 - z_i) & \forall i \\
& p_i \leq b_i + M(1 - z_i) & \forall i \\
& p_n - p_0 \leq T & \\
& z_0,\ z_n = 1 & z_i \in \{0, 1\}\, \forall i \neq 0, n \\
& y_{ij} \in \{0, 1\} & \\
& p_i \geq 0, & p_i \leq T \\
& q_{kl} \geq 0,\ r_{kij} \geq 0,\ r_{lij} \geq 0,\ t_{kij} \geq 0,\ r_{kl} \geq 0 & \forall k, l, i, j.
\end{array}
$$

□

This formulation allows every travel arc and task in the problem to potentially contribute to the execution time of any task k, subject to any additional constraints, which captures the complexity of the real-world problem. It also allows the user to allocate risk over multiple uncertain coefficients, which ameliorates the brittleness issue inherent in assigning a Γ value to only one uncertain coefficient per row of the A matrix. However, the formulation has the significant disadvantage of being intractable for any realistic-sized problem, indeed, for any problem of more than four total nodes. Specifically, this intractability is due to the dense A matrix and the large number of additional variables. In the uncertain constraints, the vast majority of the coefficients in each row are nonzero. The number of variables also increases on the order of n^2, further slowing computation time.

To illustrate these problems, the robust counterpart to one of the uncertain constraints discussed in our example is shown below. Consider again the arc (1,2) from Task 1 to Task

2, with the uncertainty sets outlined above. The original uncertain constraint is

$$p_2 - p_1 \geq \tilde{c}_{12} - M(1 - y_{12}).$$

After reformulation, this constraint takes the form

$$\begin{aligned}
&p_0^1 + \alpha_{01}^1\tilde{c}_{01} + \alpha_{02}^1\tilde{c}_{02} + \alpha_{03}^1\tilde{c}_{03} + \alpha_{12}^1\tilde{c}_{12} + \alpha_{13}^1\tilde{c}_{13} + \alpha_{21}^1\tilde{c}_{21} + \alpha_{23}^1\tilde{c}_{23} \\
&\qquad - p_0^2 + \alpha_{01}^2\tilde{c}_{01} + \alpha_{02}^2\tilde{c}_{02} + \alpha_{03}^2\tilde{c}_{03} + \alpha_{12}^2\tilde{c}_{12} + \alpha_{13}^2\tilde{c}_{13} + \alpha_{21}^2\tilde{c}_{21} + \alpha_{23}^2\tilde{c}_{23} + \tilde{c}_{12} \\
&\qquad\qquad \leq M(1 - y_{12}).
\end{aligned}$$

and its robust counterpart follows:

$$\begin{aligned}
&p_0^1 - p_0^2 + \sum_{i=0}^{2}\sum_{j=1}^{3} \alpha_{ij}^1\bar{c}_{ij} - \sum_{i=0}^{2}\sum_{j=1}^{3} \alpha_{ij}^2\bar{c}_{ij} + \bar{c}_{12} \\
&+ \Gamma_{12}q_{12} + \sum_{i=0}^{2}\sum_{j=1}^{3} r_{1ij} + \sum_{i=0}^{2}\sum_{j=1}^{3} r_{2ij} + r_{12} \\
&\qquad\qquad\qquad\qquad \leq M(1 - y_{12}) \\
&q_{12} + \sum_i\sum_j r_{1ij} \geq \hat{c}_{ij}t_{1ij} \qquad \forall i \in 0,1,2,\ j \in 1,2,3 \\
&-q_{12} + \sum_i\sum_j r_{2ij} \geq -\hat{c}_{ij}t_{2ij} \qquad \forall i \in 0,1,2,\ j \in 1,2,3 \\
&q_{12} + r_{12} \geq \hat{c}_{12} \\
&-t_{1ij} \leq \alpha_{ij}^1 \leq t_{1ij} \qquad \forall i \in 0,1,2,\ j \in 1,2,3 \\
&q_{12} \geq 0,\ r_{1ij} \geq 0,\ r_{2ij} \geq 0,\ t_{1ij} \geq 0,\ t_{2ij} \geq 0,\ r_{12} \geq 0 \quad \forall i \in 0,1,2,\ j \in 1,2,3.
\end{aligned}$$

As is apparent, this single constraint for quite a small problem has many additional variables and many nonzero coefficients, with serious implications in terms of increased running time required.

The formulation given above does have the advantage of being readily adjustable to approximation measures, such as formulating each timestamp as an affine combination of some reasonably chosen subset of travel arcs, instead of all travel arcs. The next sections propose two such approximation measures.

4.2.3 Restricted affine policies

Due to the denseness of the A matrix in the previous model and the large number of additional variables, this thesis considers two simpler affine policies.

All Incoming Arcs

With this formulation, following the convention of section 4.1.2, each uncertain constraint takes the following form:

$$\left(p_0^k + \alpha_{0k}\tilde{c}_{0k} + \alpha_{1k}\tilde{c}_{0k} + \cdots + \alpha_{(n-1)k}\tilde{c}_{(n-1)k}\right) - \left(p_0^l + \alpha_{0l}\tilde{c}_{0l} + \alpha_{1l}\tilde{c}_{0l} + \cdots + \alpha_{(n-1)l}\tilde{c}_{(n-1)l}\right) + \tilde{c}_{kl} \leq M\left(1 - y_{kl}\right).$$

After applying the same procedure demonstrated in the proof above, and the following robust counterpart is obtained:

The following is assumed: $\forall i \in \{0, ..., n-1\}$, $j \in \{1, ..., n\}$, $\forall k = 0, ..., n-1, l = 1, ..., n$.

$$\begin{array}{lll}
\text{maximize} & \sum_{i=1}^{n} f_i(z_i) & \\
\text{subject to} & z_0,\ z_n = 1 & z_i \in \{0,1\}\, \forall i \neq 0, n \\
& y_{ij} \in \{0,1\} & \\
& y_{ii} = 0 & \forall i \in \{1, ..., n-1\} \\
& \sum_{j=1}^{n} y_{ij} = z_i & \forall i \in \{0, ..., n-1\} \\
& \sum_{i=0}^{n-1} y_{ij} = z_j & \forall j \in \{1, ..., n\} \\
& p_0^k - p_0^l + \sum_i \alpha_{ik}\bar{c_{ik}} - \left(\sum_i \alpha_{il}\bar{c}_{il}\right) + \bar{c}_{kl} & \\
& \quad + \Gamma_{kl}q_{kl} + \sum_i r_{ik} + \sum_i r_{il} + s_{kl} & \leq M\left(1 - y_{kl}\right) \\
& q_{kl} + \sum_i r_{ik} \geq \hat{c}_{ij}t_{ik} & \forall i, k, l, i \neq k \\
& -q_{kl} + \sum_i r_{il} \geq -\hat{c}_{il}t_{il} & \forall i, k, l, i \neq k \\
& q_{kl} + r_{kl} \geq \hat{c}_{kl} &
\end{array}$$

$$\begin{aligned}
-t_{ik} \leq \alpha^*_{ik} \leq t_{ik} & \qquad \forall k, i, i \neq k \\
p_j - p_i \geq l_{ij} - M(2 - z_i - z_j) & \qquad \forall i, j \\
p_j - p_i \leq u_{ij} + M(2 - z_i - z_j) & \qquad \forall i, j \\
p_i \geq a_i - M(1 - z_i) & \qquad \forall i \\
p_i \leq b_i + M(1 - z_i) & \qquad \forall i \\
p_n - p_0 & \qquad \leq T \\
p_i \geq 0, & \qquad p_i \leq T \\
q_{kl} \geq 0,\ r_{ik} \geq 0,\ t_{ik} \geq 0,\ s_{kl} \geq 0 & \qquad \forall k, l, i, j.
\end{aligned}$$

This formulation does not exactly replicate the full structure of the problem; as discussed above, it assumes that the immediately previous travel arc is the largest influence on the p_0^k decision variable. However, this modeling decision also reflects reality, in that the uncertainty between the previous timestamp and the current timestamp stems totally from whichever travel arc lies between the two relevant nodes. Therefore it is reasonable to assume that this formulation, while not encompassing the full scope of the problem, nevertheless does not artificially restrict the path-planning of the model.

Computationally, this formulation significantly decreases both the number of nonzero coefficients in the A matrix and the number of variables in the problem. These simplifications improve its tractability over the full affine formulation.

h-Nearest Neighbors

Without rehearsing again the mechanics of arriving at the appropriate constraints (similar to the All Incoming Arcs policy, but restricted) the final form of this robust counterpart follows:

The following is assumed: $\forall k = 0, ..., n-1, l = 1, ..., n$.

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} f_i z_i & \\
\text{subject to} \quad & z_0,\ z_n = 1 & z_i \in \{0,1\} \forall i \neq 0, n \\
& y_{ij} \in \{0,1\} & \\
& y_{ii} = 0 & \forall i \in \{1, ..., n-1\}
\end{aligned}$$

$$
\begin{aligned}
\sum_{j=1}^{n} y_{ij} &= z_i && \forall i \in \{0, ..., n-1\} \\
\sum_{i=0}^{n-1} y_{ij} &= z_j && \forall j \in \{1, ..., n\} \\
p_{k0} - p_{l0} + \sum_i \alpha_{ik}\bar{c_{ik}} - \left(\sum_j \alpha_{jl}\bar{c_{jl}}\right) + \bar{c_{kl}} & && \\
+ \Gamma_{kl} q_{kl} + \sum_i r_{ik} + \sum_i r_{il} + s_{kl} & && \leq M\left(1 - y_{kl}\right) \forall i \in H_k, \forall j \in H_l \\
q_{kl} + \sum_i r_{ik} \geq \hat{c_{ij}} t_{ik} & && \forall i, k, l, i \neq k \\
-q_{kl} + \sum_i r_{il} \geq -\hat{c_{il}} t_{il} & && \forall i, k, l, i \neq k \\
q_{kl} + r_{kl} \geq \hat{c_{kl}} & && \\
-t_{ik} \leq \alpha^*_{ik} \leq t_{ik} & && \forall k, i, i \neq k \\
p_j - p_i \geq l_{ij} - M(2 - z_i - z_j) & && \forall i, j \\
p_j - p_i \leq u_{ij} + M(2 - z_i - z_j) & && \forall i, j \\
p_i \geq a_i - M(1 - z_i) & && \forall i \\
p_i \leq b_i + M(1 - z_i) & && \forall i \\
p_n - p_0 & && \leq T \\
p_i \geq 0, & && p_i \leq T \\
q_{kl} \geq 0,\ r_{ik} \geq 0,\ t_{ik} \geq 0,\ s_{kl} \geq 0 & && \forall k, l, i, j.
\end{aligned}
$$

where H_k is the set of h nodes closest to a given node k and H_l is the set of h nodes closest to a given node l.

This method again decreases significantly the number of nonzero coefficients in the A matrix from the All Incoming Arcs method. Its tractability is improved over the All Incoming Arcs method, as shown in the next chapter.

Chapter 5

Computational Results

This chapter demonstrates that the methods detailed above can solve problems of a realistic size in a reasonable running time, showing the tractability of the methods. It further demonstrates the quality of these solutions through simulation, by comparing both failure rates and expected reward (objective function value) of the All Incoming Arcs method (4.2.3) and the h-Nearest Neighbor method (4.2.3) versus the deterministic method (3) and the naive method (4.2.1). Additionally this chapter shows that most of the improvement in objective function value is achieved in the first quintile of time required to find the provably optimal objective function value. This last observation supports the tractability of the approaches presented in the thesis.

Sample problems were randomly generated ranging in size from 12 tasks (including start and end) to 32 tasks. Fifty problems were generated of each size. For each problem, a list of n latitude-longitude pairs was randomly chosen (where n is the number of tasks) and its corresponding list of travel times between each pair of tasks. Rewards were then randomly assigned to each task, excepting the start and end tasks, which had reward 0 for every problem. The upper bound T was randomly chosen, choosing from a user-defined range that spanned very loose bounds to quite tight bounds. For the robust problems, maximum divergences were randomly generated $\hat{c}_{ij}$ from $\bar{c}_{ij}$ for each arc (i, j). These divergences were between 5 and 20 percent of $\bar{c}_{ij}$. Finally, the Γ value for each row of the robust problems was randomly selected from a user-chosen range.

The problems were run on a MacBook Pro OS 10.6.8, with a 2.4 GHz Intel Core Duo processor and 4GB of RAM, and the runtime is reported in seconds. The MILPs were optimized using PuLP [1] as a modeler and Gurobi [14] as the solver.

Each solution was tested by simulation, using a simulator in Python which randomly generates a speed-of-advance for the vehicle at each 1-minute timestep. This speed of advance is used to generate the simulated distance traveled by the vehicle during that minute. This distance traveled is then compared to the distance traveled required by the solution being tested to get to the next task on time. If the simulated progress of the vehicle falls too far behind the solution being tested to make it to the next task on time, then the simulator records that event as a failure of the solution being tested. A buffer value of 30 percent of the average distance between tasks was used to mean too far behind. The maximum possible divergence from the nominal travel time used to calculate the speed-of-advance is 30 percent of the nominal travel time, a worse-case scenario than the robust problems assumed. The simulator also records the point in elapsed time at which each failure happened.

5.1 Runtime results

Table 5.1 and Table 5.2 present the mean and quartiles for 50 random problems of the given size. In the following tables, the heading "Det." refers to the deterministic problem, "All" refers to the All Incoming Arcs method, and the heading "h-NN" refers to the h-Nearest Neighbors method.

	Det.		Naive		All		h-NN	
Size	Mean	SD	Mean	SD	Mean	SD	Mean	SD
12	0.02	0.02	0.02	0.02	3.26	6.07	0.82	1.06
17	0.14	0.20	0.16	0.26	389	1040	242	781
22	0.18	0.17	0.12	0.10	532	1140	56.9	83.3
27	0.24	0.29	0.25	0.21	2630	3530	203	234
32	1.36	1.88	1.09	0.92	21700	19100	1820	3070

Table 5.1: Mean and standard deviation, reported in seconds, of runtime of various problem sizes. "All" indicates the All Incoming Arcs method, and "h-NN" indicates the h-Nearest Neighbors method.

Table 5.1 shows that the deterministic method is consistently very fast; the All Incoming Arcs method is the slowest, while the h-Nearest Neighbors method tends to perform one order of magnitude better than the All Incoming Arcs method. The h-Nearest Neighbors method averages half an hour for the largest problems. However, the means are somewhat skewed by outlying problems, as indicated by typically large standard deviations. Table 5.2 shows that the robust methods in particular perform better than the mean would indicate. The median for the largest problems for the h-Nearest Neighbors method is ten minutes, instead of half an hour. These results also show that the deterministic method performs better than its means would indicate.

	Det.			Naive			All			h-NN		
Size	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
12	0.01	0.01	0.01	0.01	0.01	0.01	0.38	0.95	2.44	0.13	0.59	0.87
17	0.02	0.09	0.19	0.03	0.06	0.18	18.4	25.9	198	3.10	8.22	49.5
22	0.06	0.11	0.23	0.07	0.11	0.13	68.3	164	303	11.0	20.8	65.1
27	0.09	0.10	0.21	0.13	0.13	0.23	534	1040	3540	42.2	141	273
32	0.30	0.85	1.63	0.35	0.74	1.51	8960	13800	29500	491	601	1960

Table 5.2: First, second, and third quartiles, reported in seconds, of runtime of various problem sizes. "All" indicates the All Incoming Arcs method, and "h-NN" indicates the h-Nearest Neighbors method.

Figure 5-1 shows the median runtimes of the four solution methods. The behavior of the runtimes indicates that the robust methods have the potential to scale relatively well, especially given the behavior in Figures 5-2 and 5-3.

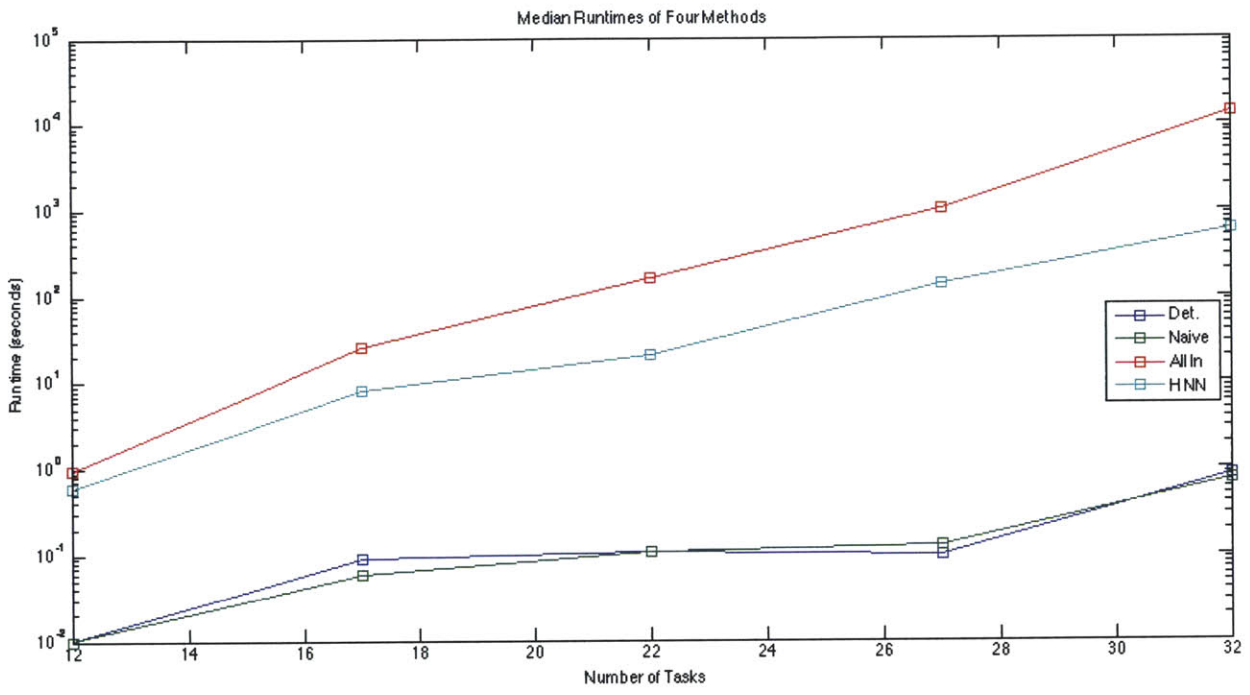

Figure 5-1: Median runtime shown graphically for the deterministic, naive, All Incoming Arcs, and h-Nearest Neighbors methods.

Figures 5-2 and 5-3 demonstrate that, even though the computation time is long for the h-Nearest Neighbors policy, the significant majority of solution improvement is achieved very early in terms of computation time required to produce the optimal solution with a tolerance of 1×10^{-10}; the remaining time required has immensely diminished rewards. This pattern holds for the deterministic case as well.

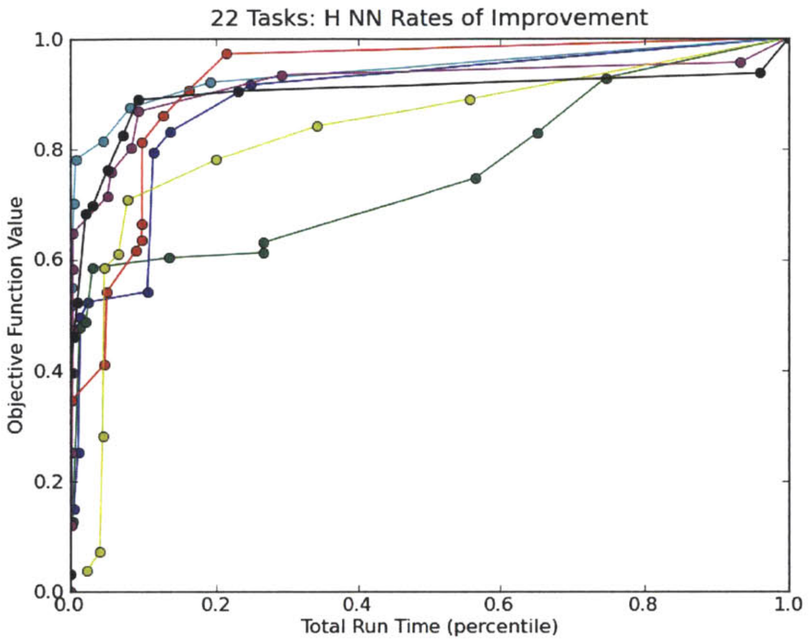

Figure 5-2: Rate of improvement shown for seven randomly selected problems with 22 tasks, with the h-Nearest Neighbors method. Each color indicates a separate problem instance.

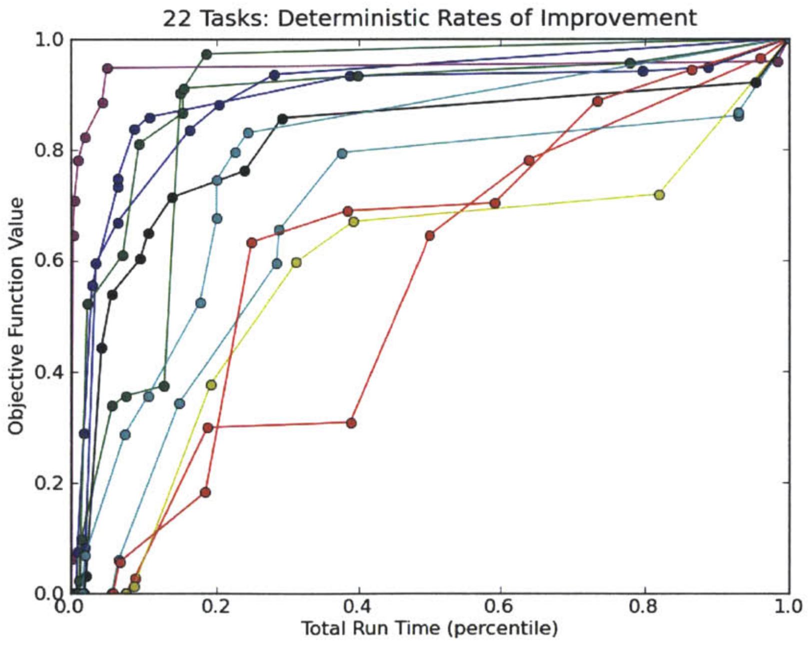

Figure 5-3: Rate of improvement shown for eleven randomly selected problems with 22 tasks, with the deterministic method. Each color indicates a separate problem instance.

5.2 Solution quality results

This section presents the results of simulating solutions. Figure 5-4 illustrates the process of simulation described at some length at the beginning of this section.

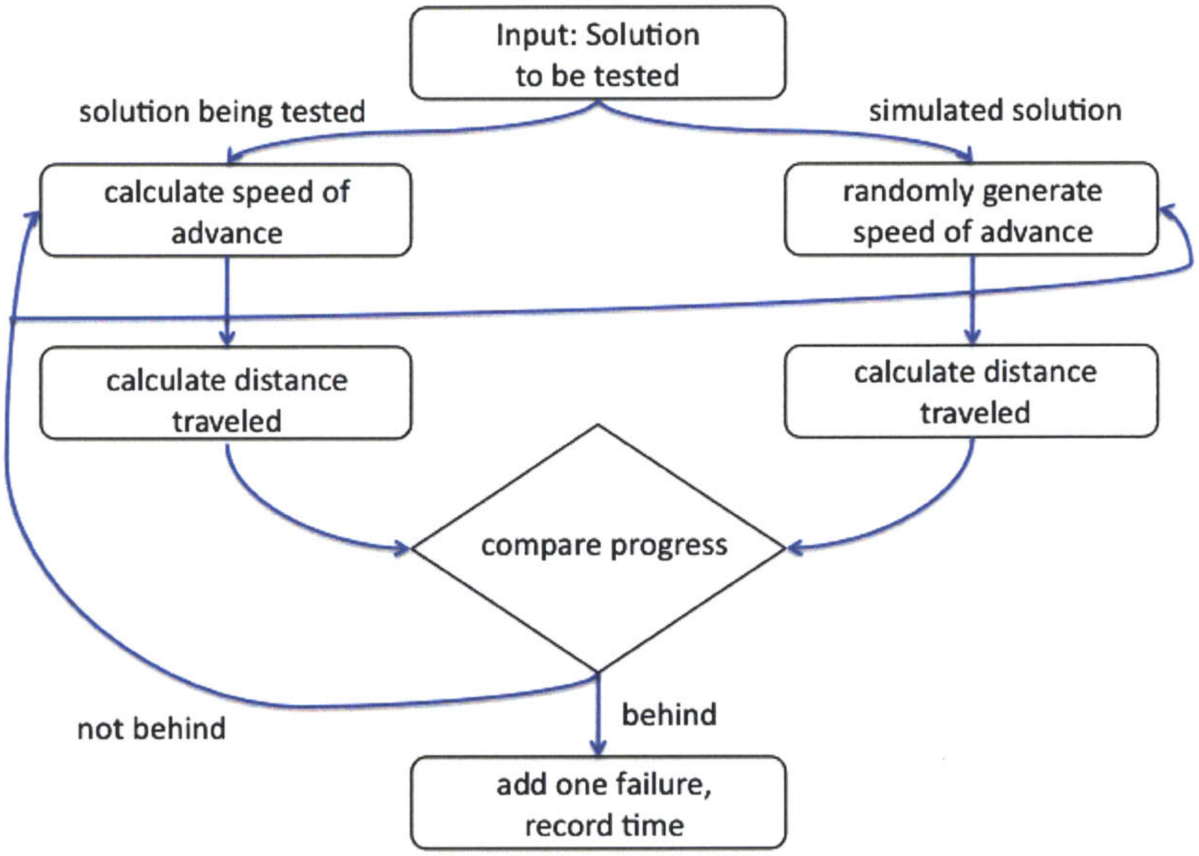
Input: Solution to be tested
solution being tested
simulated solution
calculate speed of advance
randomly generate speed of advance
calculate distance traveled
calculate distance traveled
compare progress
not behind
behind
add one failure, record time

Figure 5-4: This flowchart shows the simulation process. Arcs represent information being passed, blocks represent executable steps. The process takes place once a simulated minute.

Table 5.3, Table 5.4, and Figure 5-5 show the expected reward for each method for each task size, based on simulation. The objective function value of this problem is to be maximized. The expected reward for each problem has been normalized to a maximum value of 1, to allow for side-by-side comparison.

	Det.		Naive		All		h-NN	
Size	Mean	SD	Mean	SD	Mean	SD	Mean	SD
12	0.51	0.12	0.51	0.08	0.98	0.05	0.98	0.05
17	0.46	0.13	0.46	0.12	0.92	0.27	0.97	0.14
22	0.40	0.09	0.43	0.13	0.99	0.04	0.99	0.04
27	0.42	0.16	0.41	0.16	0.94	0.16	0.97	0.14
32	0.41	0.17	0.33	0.08	0.99	0.03	0.92	0.25

Table 5.3: Normalized expected reward (objective function value) for each method for each task size, based on simulation.

	Det.			Naive			All			h-NN		
Size	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
12	0.45	0.50	0.53	0.47	0.51	0.56	0.97	1	1	1	1	1
17	0.40	0.44	0.52	0.38	0.43	0.51	1	1	1	1	1	1
22	0.35	0.38	0.46	0.35	0.40	0.48	1	1	1	1	1	1
27	0.32	0.38	0.45	0.30	0.38	0.45	0.98	1	1	1	1	1
32	0.29	0.35	0.49	0.27	0.33	0.38	1	1	1	0.98	1	1

Table 5.4: First, second, and third quartiles of expected reward normalized to 1.

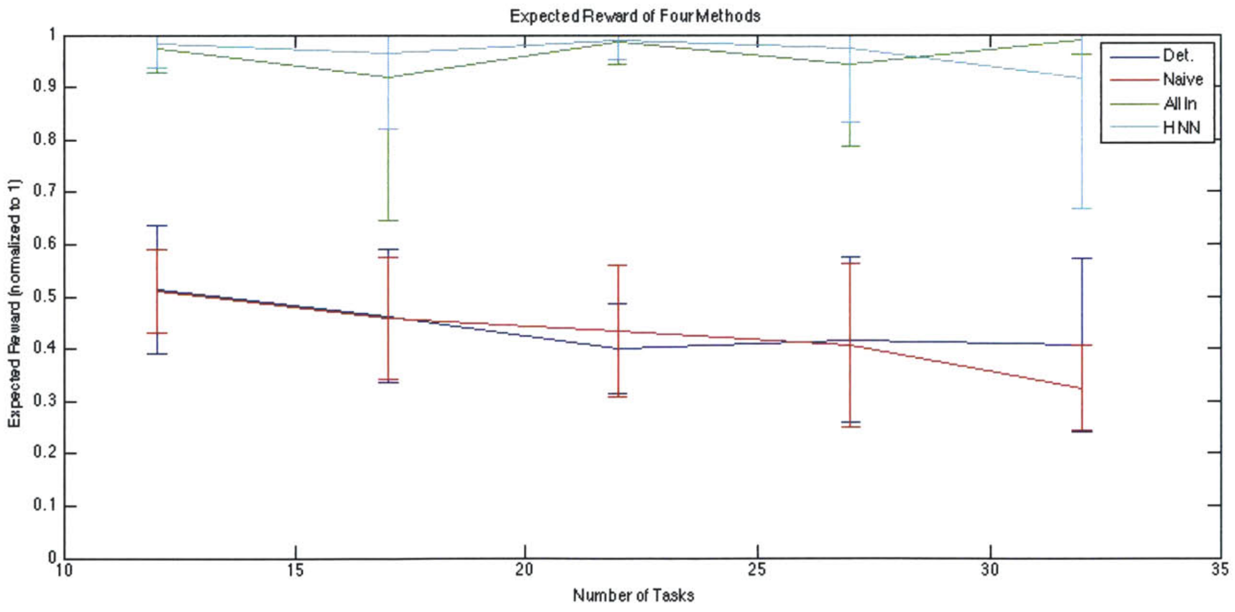

Figure 5-5: Normalized expected reward for each method for each task size, with standard deviation bars.

While it appears that the h-Nearest Neighbors method performs better than the All Incoming Arcs method, the standard deviations demonstrate that this difference is not significant.

Table 5.3 and Figure 5-5 demonstrate that the robust methods substantially outperform the deterministic method in terms of expected reward. Notably, the standard deviation bars do not overlap. Table 5.4 shows that the first, second, and third quartiles of expected reward for our robust methods clusters very closely to the maximum possible reward. The expected reward for the deterministic method does not deliver the same value.

The next tables show the results of simulation in terms of failures and failure points. Each solution was simulated 200 times. Table 5.5 and Table 5.6 present the mean and quartiles for 50 problems of the given size. A mean of 25, for example, indicates that there were on average 25 failures out of 200 simulations for that task size, averaged over 50 problems.

	Det.		Naive		All		h-NN	
Size	Mean	SD	Mean	SD	Mean	SD	Mean	SD
12	97.4	24.5	98.0	16.0	5.06	9.41	3.18	9.37
17	107	25.7	108	23.3	16.3	54.1	6.78	28.8
22	120	17.3	113	25.0	2.82	8.75	2.31	7.00
27	117	31.6	119	31.2	11.3	31.1	5.41	28.3
32	119	33.3	135	15.9	2.06	5.64	16.9	49.7

Table 5.5: Mean and standard deviation of number of failures over 50 problems of each size, simulated 200 times each. The average number of failures is out of 200 trials.

	Det.			Naive			All			h-NN		
Size	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
12	94	101	111	88.5	99	107	0	0	7	0	0	0
17	96.5	113	121	98.5	115	124	0	0	0	0	0	0.5
22	109	124	131	104	120	130	0	0	0	0	0	0
27	110	124	136	110.5	125	140	0	0	5	0	0	0
32	102	131	142	124	135	147	0	0	0	0	0	3.75

Table 5.6: First, second, and third quartiles of number of failures over 50 problems of each size, simulated 200 times each. The number of failures is out of 200 trials.

Table 5.5 and Table 5.6 indicate that the deterministic solutions fail very frequently, over half the time even when considering quartiles instead of the mean. On the other hand, the robust solutions fail very infrequently, less than ten percent of the time in the worst case (the h-Nearest Neighbors mean for 32 tasks). The quartile results for the robust methods indicate that typically, these solutions fail extremely infrequently in simulation.

Table 5.7 and Table 5.8 below illustrate the points of failure for each solution size. The number reported indicated the percentile of total solution time T at which the solution being tested failed, if it failed. Table 5.7 and Table 5.8 present the mean and quartiles for the points of failure for each of 200 simulations for each of the 50 problems for a given size.

	Det.		Naive		All		h-NN	
Size	Mean	SD	Mean	SD	Mean	SD	Mean	SD
12	0.20	0.13	0.20	0.12	0.17	0.12	0.09	0.12
17	0.22	0.18	0.23	0.19	0.17	0.05	0.07	0.08
22	0.20	0.17	0.18	0.17	0.08	0.04	0.10	0.07
27	0.17	0.17	0.17	0.17	0.12	0.08	0.08	0.02
32	0.18	0.19	0.19	0.19	0.07	0.04	0.19	0.13

Table 5.7: Mean and standard deviation of failure points over 50 problems of each size, simulated 200 times each.

	Det.			Naive			All			h-NN		
Size	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
12	0.11	0.17	0.26	0.11	0.17	0.26	0.07	0.14	0.26	0.01	0.03	0.16
17	0.08	0.18	0.30	0.09	0.17	0.31	0.12	0.18	0.20	0.02	0.03	0.11
22	0.08	0.15	0.28	0.07	0.12	0.22	0.05	0.07	0.09	0.05	0.07	0.12
27	0.06	0.11	0.23	0.05	0.10	0.21	0.05	0.09	0.22	0.07	0.08	0.09
32	0.05	0.11	0.23	0.06	0.12	0.26	0.04	0.08	0.09	0.05	0.25	0.29

Table 5.8: First, second, and third quartiles of failure points over 50 problems of each size, simulated 200 times each.

Table 5.7 and Table 5.8 indicate that solutions of all three types tend to fail early in the solution timeline, when they fail. This tendency is more pronounced for the robust methods, but even the third quartile for the deterministic method indicates that failures take place in the first 30 percent of the solution timeline. Figures 5-6, 5-7, and 5-8 demonstrate this tendency visually. This pattern holds for other problem sizes.

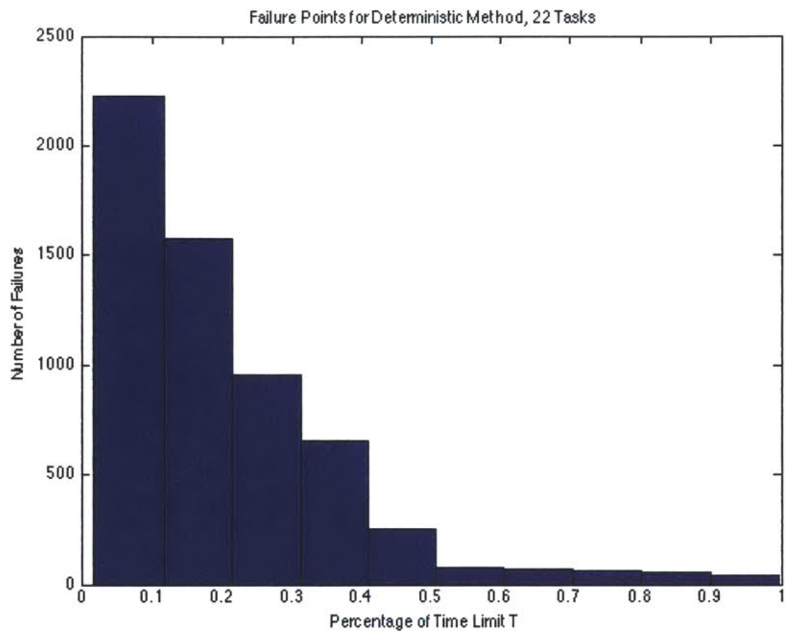

Figure 5-6: Number of failures in each decile of the overall solution time for the deterministic method, size 22 tasks.

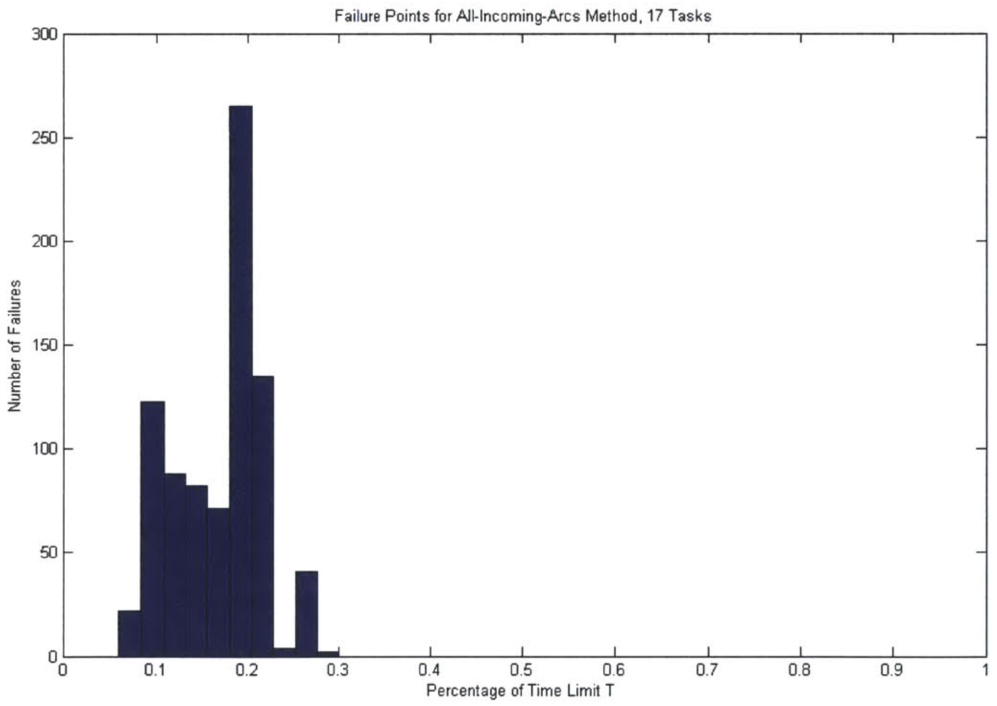

Figure 5-7: Number of failures in each decile of the overall solution time for the All Incoming Arcs method, size 17 tasks.

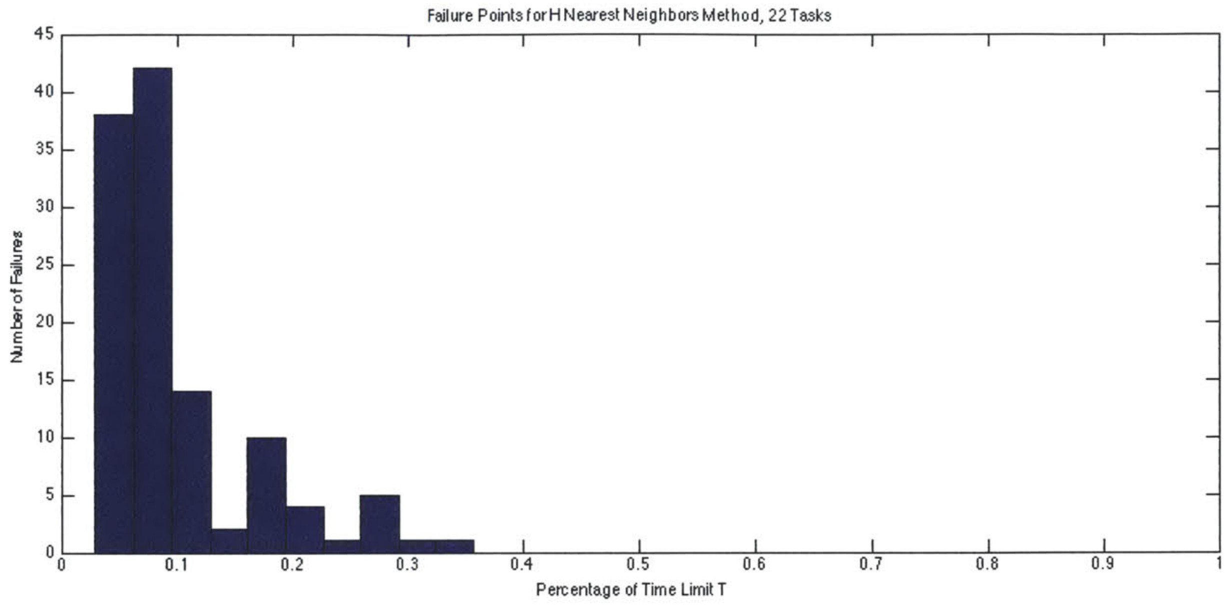

Figure 5-8: Number of failures in each decile of the overall solution time for the h-Nearest Neighbors method, size 22 tasks.

Figure 5-9 demonstrates that, when h-Nearest Neighbor solutions fail, they fail later in the solution timeline as the problem size grows. The opposite pattern holds true for deterministic solutions. This tendency suggests that the h-Nearest Neighbor method becomes, if anything, slightly more robust with larger problem sizes. This data is also reported in Table 5.8.

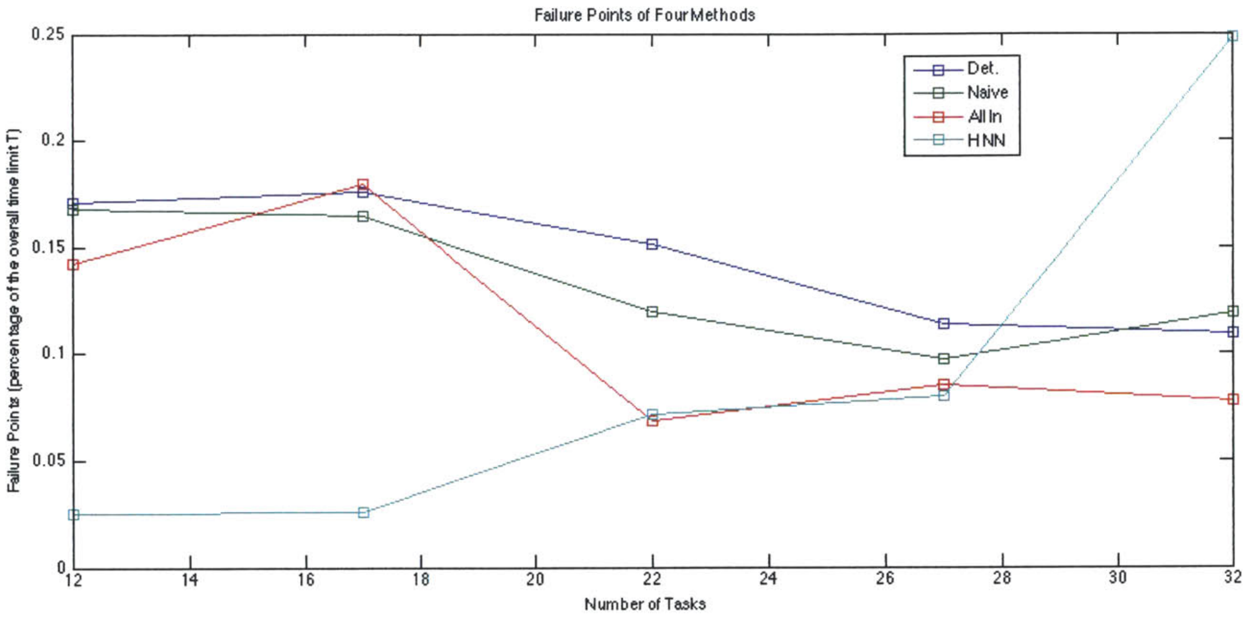

Figure 5-9: Median failure points for the deterministic, naive, All Incoming Arcs, and h-Nearest Neighbors methods.

These results demonstrate that these methods can solve problems of up to 32 tasks in a reasonable runtime. Simulation demonstrates that using robust optimization improves the quality of the solution in terms of likelihood that the optimal plan yielded by the model would succeed in an uncertain environment. Simulation also demonstrates that the robust policies perform much better in terms of expected reward than the deterministic method, and furthermore that the robust policies yield expected reward very close to the maximum reward possible.

Chapter 6

Conclusions

This thesis presented a novel method for robust route optimization for unmanned underwater vehicles, wherein each key decision variable (each timestamp) is reformulated as an affine combination of all travel arcs in the planning problem. It also presented two restricted versions of this method. The All Incoming Arcs approach formulated each timestamp as an affine combination of the travel arcs flowing into that timestamp's node. This approach was more tractable than the full affine method, being able to handle up to 22 tasks in a reasonable runtime, but it was much less tractable than the deterministic problem. The h-Nearest Neighbors method formulated each timestamp as an affine combination of the travel arcs flowing into that node from that node's h nearest neighbors. This method was much more tractable than the All Incoming Arcs method, but still less tractable than the deterministic formulation. As shown in the computational results, the h-Nearest Neighbors method could handle up to 32 tasks in a reasonable runtime; larger problems were not tested for this paper.

Despite the disadvantages of the restricted affine policies, they do provide highly robust solutions which supply ample buffer time between nodes to protect against uncertain events such as tides, currents, and surface traffic, among others.The h-Nearest Neighbors method, in particular, achieves nearly 100% of expected reward in simulation, with a typical failure rate of under 5%, suggesting increased robustness. Unlike previous robust attempts like the naive robust approach, it is not brittle; it also enjoys some computation time advantages over prior art in the area of robust solutions (see [9, 10]). There is certainly room for improvement, which should be sought in other heuristics which further decrease the number of nonzero coefficients in the A matrix without compromising the scope of the model.

Future work might include further refining a restricted affine policy of the kind presented in this thesis; it may also include a multistage adaptive affine policy, wherein the problem is optimized over multiple time stages. Finally, future work could involve including Crimmel's work on navigation points. Adding navigation points to the mix of path planning constitutes another layer of complexity, and may require other approaches.

Bibliography

[1] PuLP: An LP modeler in Python, 2013. code.google.com/p/pulp-or/.

[2] E. Guslitzer, A. Ben-Tal, A. Goryashko and A. Nemirovski. Adjustable robust solutions of linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[3] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13, 1999.

[4] A. Ben-Tal and A. Nemirovski. Robust optimization – methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.

[5] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Nashua, NH, 1999.

[6] D. Bertsimas and V. Goyal. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical Programming*, 2009.

[7] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(2):35–53, 2004.

[8] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Nashua, NH, 1997.

[9] J. Cates. Route optimization under uncertainty for unmanned underwater vehicles. Master's thesis, Massachusetts Institute of Technology, 2011.

[10] B. Crimmel. A priori and on-line route optimization for unmanned underwater vehicles. Master's thesis, MIT, 2012.

[11] D. Brown, D. Bertsimas and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53:464–501, 2011.

[12] D. Iancu, D. Bertsimas and P. Parrilo. Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394, 2010.

[13] ADM Greenert. CNO's Sailing Directions, 2012. www.navy.mil/cno/cno_sailing_directions_final_lowres.pdf.

[14] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2012. http://www.gurobi.com.

[15] A. Martin. U.S. Expands Use of Underwater Unmanned Vehicles. *Undersea Warfare*, April 2012. www.nationaldefensemagazine.org/archive/2012/April/Pages /USExpandsUseOfUnderwaterUnmannedVehicles.aspx.

[16] D. Pachamanova. *A robust optimization approach to finance.* PhD thesis, MIT, 2002.

[17] United States Navy. United States Navy Unmanned Underwater Vehicles Master Plan. April 2004. http://www.navy.mil/navydata/technology/uuvmp.pdf.

[18] E.C. Whitman. Unmanned Underwater Vehicles: Beneath the Wave of the Future. *Undersea Warfare*, 2002. www.navy.mil/navydata/cno/n87/usw/issue_15/wave.html.