PRACTICAL MILITARY APPLICATIONS OF TIMETABLING, PATH PLANNING, AND TIME-VARYING NETWORKS FOR MAXIMIZING MISSION SUCCESS AND MINIMIZING RISK

by

Christopher D. Richards

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Interdisciplinary).

Golden, Colorado

Date _____

Signed: _____

Christopher D. Richards

Signed: _____

Dr. Alexandra M. Newman

Thesis Advisor

Signed: _____

Dr. Dinesh P. Mehta

Thesis Advisor

Golden, Colorado

Date _____

Signed: _____

Dr. Gregory Jackson

Professor and Head

Department of Mechanical Engineering

ABSTRACT

This dissertation consists of three papers in various stages of publication. The first paper, "Optimal Course Scheduling for United States Air Force Academy Cadets" is currently in the final stage of review by the journal *Interfaces*. The paper presents a mixed integer program that considers faculty preferences along with cadet military, athletic, and academic requirements for on-time commissioning. The model is developed as part of an iterative process between faculty and the registrar to build course timetables and student schedules that honor the limited university resources and manpower. The second paper, "Minimum-Risk Routing Through a Mapped Minefield," has been submitted and is currently under review by *Networks*. We build a "threat-additive" approach to measuring the risk incurred by a ship navigating a naval minefield. We employ this approach, both as an integer program and as an A* search, to identify minimum-risk paths that may not be found by traditional "edge-additive" models. The last paper, "Navigating Free-Floating Minefields via Time-Evolving Voronoi Graphs," is ready for submission pending journal-specific formatting. We use environmental data to generate Voronoi graphs representing static "snapshots" of a free-floating minefield over time. Using these snapshots, we build a time-varying graph through which parametrically defined minimum-risk journeys can be found.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

In recent decades, there a gap has grown between the mission the Department of Defense (DoD) is expected to accomplish and the tools it is provided to do so. During a congressional hearing on the future of of the Army, Congressman Michael Turner argued that "The Army is being asked to do more with less." (Turner 2016). This is not a new trend; in the years immediately following the Korean, Vietnam, and Cold Wars, the U.S. enforced austere 30% to 40% budget reductions (Plumer 2017). In the years since major operations in Afghanistan and Iraq, this trend has reasserted itself, with many people going as far to argue that support for major weapons systems be discontinued (Plumer 2017), military academies be closed (Fleming 2015), and entire branches of the DoD be dismantled (Tedeschi 2015). As an alternative to such reactionary measures, this dissertation considers the question of how Operations Research (OR) techniques can be applied to reduce the time, resources, and manpower required for several military capabilities currently facing scrutiny and potential drawdown.

We begin with an analysis of the commissioning process at the United States Air Force Academy (USAFA), where discontinuation of their cadet management system requires that they find a replacement for their current scheduling software. We assist by focusing on solving the Population Complete Timetabling Problem at USAFA. With over 4,000 cadets and approximately 400 courses, the Air Force spends hundreds of manhours and millions of dollars each year to develop course offerings and class schedules that ensure military commissioning deadlines are met. The model we develop uses a mixed-integer program to incorporate faculty preferences with cadet registrations to efficiently build schedules that meet the military, athletic, and academic requirements of a commission in the United States Air Force. Unlike the scheduling process USAFA has followed for over fifty years, that necessitates a unique alternating-day class schedule, our model finds feasible schedules that employ the standard repeated-week format common to most commercial-off-the-shelf

university timetabling software. In contrast to the previous USAFA process, we greatly reduce the amount of time spent finding schedules, respect 90% of faculty preferences, use 21% fewer sections, provide much needed measures of solution quality, and successfully save USAFA over $120 million in customization costs.

Next, we examine the challenge faced by U.S. Naval forces with respect to mine warfare. With financial support dwindling for ships with mine-clearing capabilities (McNally 2012), we explore strategies for mapping minefields and measuring the minimum risk incurred by ships traversing them. In order to identify minimum-risk paths, we integrate this new "threat-additive" approach into two separate integer programs. When compared to traditional "edge-additive" approaches to measuring risk, the integer programs we develop find lower-risk paths in 43% of test cases in an average of 90 seconds or less. In addition to these integer programs, we develop a simple, open-source piece of software that facilitates ease-of-use by Naval forces. While this software uses an A* search and possesses exponential worst-case runtimes, we introduce several heuristics that identify initial solutions and greatly reduce the number of paths that must be evaluated. Using these heuristics, our A* search algorithm finds paths with an average of 15% less risk than traditional models in an average of one second.

Lastly, we build on our work regarding naval minefields to address the growing concern posed to military forces by Free-Floating Minefields (FFMs). Like IEDs, FFMs have grown in popularity as a weapon of choice during asymmetric warfare, despite being condemned by the international community. In contrast to conventional minefields, mines within an FFM are untethered and severely complicate navigation due to the drift that current and wind slippage can induce. To account for these disruptions, we use environmental data, along with oceanic drift models, to generate static "snapshops" of the area of operations (AO). These snapshots build a time-varying graph (TVG) that represents paths both spatially and temporally. Unlike previous research, which focuses either on the the theoretical characteristics of Voronoi-based TVGs or on how the partitions they generate can be prac-

tically applied, we develop a specific series of equations and algorithms that define each edge parametrically, and ensure connectivity in order to realistically describe the motion of a ship navigating an FFM of any size and shape. We then modify our A* algorithm from our previous paper to quickly identify minimum-risk "journeys" through the TVG. Using real-world data collected from National Oceanic and Atmospheric Administration buoys, we show that the journeys we find are well within the capabilities of a naval ship navigating a minefield, and present several methods to increase the tractability of TVG generation.

CHAPTER 2

OPTIMAL COURSE SCHEDULING FOR UNITED STATES AIR FORCE ACADEMY CADETS

A paper under second review by the journal *Interfaces*.

Gerardo Gonzalez [1] [2], Christopher Richards [3] [4], Alexandra Newman [5]

2.1 Abstract

Scheduling students and academic courses at the United States Air Force Academy (USAFA), a military commissioning source, has required unique software and considerable manual effort. The recent discontinuation of the Oracle-based student information system mandates that the Superintendent invest in new software, the customization of which will incur millions in additional costs if USAFA continues to rely upon a fixed alternating-day schedule format. We present an integer program that generates a course schedule using the repeated-week format common to most commercial-of-the-shelf (COTS) systems. The integer program uses cadet registration information to determine the number of sections to be offered and how cadets should be assigned to them in order to ensure on-time graduation while accomplishing mandatory military training. Hard constraints enforce institutional restrictions that require all athletes to attend practice, limit the number of cadets who delay required courses, keep classroom usage and number of sections to campus and faculty availability, and ensure cadets are assigned only to scheduled sections without overlapping time requirements. Flexible constraints reflect faculty and cadet preferences; their violation is minimized to honor teaching requests from each department, maintain minimum and maximum section sizes, restrict the number of evening sections, and meet cadet registrations. In contrast to the previous USAFA process, we generate schedules that reduce the number

[1]Lieutenant Colonel, United States Air Force.
[2]Assistant Professor, Department of Management, Unted States Air Force Academy.
[3]Major, United States Air Force.
[4]PhD Candidate, Mechanical Engineering Department, Colorado School of Mines.
[5]Professor, Department of Mechanical Engineering, Colorado School of Mines.

of unmet student registrations by more than 75%, use 21% fewer sections, and respect nearly 90% of faculty teaching preferences. Results from our methodology are easily reproducible and measurable in terms of time to adjudicate, desirability, and demand on faculty resources. By accommodating a standard repeated-week format, rather than adhering to the current alternating-day approach, our model integrates easily as a front-end to a COTS system and avoids $120 million in customization costs. Our program reduces the reliance on manual manipulation and makes it possible to find feasible schedules that permit section length and patterns to vary according to pedagogy—a break from over fifty years of rigid time-blocking techniques that sacrifice desirability for feasibility and timeliness.

2.2 Introduction

Established in 1954, the United States Air Force Academy (USAFA) is a public four-year university, comprised of 25 academic departments, with an approximate enrollment of 4,400 undergraduates pursuing over 30 possible degrees. As a military academy, however, USAFA differs from other four-year universities in four key aspects: (*i*) cadets have military jobs and duties that require a significant portion of their day to accomplish; however, some commitments may last only a portion of a semester, leaving a gap unsuitable for a standard academic class; (*ii*) USAFA requires at least one airmanship course (i.e., flying gliders, parachuting out of aircraft, or flying motorized airplanes) that requires airfield access and a larger portion of the day than a standard academic course; (*iii*) USAFA requires 100% cadet involvement in athletic training. Specifically, about 25% of the cadet body participates in an intercollegiate sport, while the remainder participates in intramural sports; and (*iv*) with taxpayer-funded tuition and military quotas designed to be consistent with predetermined commissioning dates, cadets must graduate within four years.

For cadets to meet the competing requirements of academics, military training and athletics, the registrar must necessarily consider cadet registrations each semester to simultaneously build course schedules and cadet assignments. When the registrar starts her process, she cannot gauge the number of cadet registrations her initial schedule will fail to

fulfill, the work required to fix them, or if they can even be fixed at all. To plan for the worst case, the registrar must either collect information regarding cadet-course preferences and department-course offerings months prior to the semester being scheduled, or release cadet schedules for the next semester very late in the current semester. The former strategy precludes cadets and department staff from thoroughly considering the following semester's academic requirements, while the latter leaves insufficient time to resolve conflicts that may arise.

The majority of the information concerning cadet status and resource allocation is tracked by a collection of applications called the Cadet Administrative Management Information System (CAMIS), designed specifically to accommodate USAFA's unique scheduling requirements by implementing strict, uniform, block-like timetables. This structure simplifies the manual manipulation in which departments and the registrar must engage each semester in order to resolve conflicts in a timely manner. Efforts to modernize CAMIS have consumed more than $30 million over the past decade, culminating in missed milestones and cost overruns (Air Force A3/5 2017). As a result, Headquarters Air Force has advised that all further attempts at modernization must rely heavily on commercial off-the-shelf (COTS) systems with minimal customization. Recently, the software that supports CAMIS has been discontinued, and commercial support has been withdrawn. As a result, USAFA must make the critical decision on how to replace its student information system. In order to adopt COTS software, USAFA must either have the system customized to accommodate the unique timetable structure it currently uses, or develop the ability to generate schedules which adapt to the standard repeated-week format common to COTS systems while still meeting USAFA requirements.

Acting on the recommendations of a 2012 report (HQ USAFA 2012), Superintendent Lieutenant General Michelle Johnson mandated that a "Pathways" Committee develop a model capable of generating repeated-week schedules. To be implemented, such a model must maintain the emphasis on academic, military and athletic requirements, but also

introduce flexibility with additional section lengths and meeting patterns. Rather than remove departments and the registrar from the process, input from both should determine how to honor departmental preferences while respecting institutional requirements. This approach allows departments to focus on arranging courses based on what they deem pedagogically sound (e.g., whether a particular course is more suited as a once-a-week lecture or a thrice-a-week practicum). Due to the number and variety of stakeholders involved in such a model, USAFA leadership agrees with Schaerf (1999) that such a timetabling problem cannot and should not be completely automated. Any model, therefore, must remain part of an iterative process between departments and the registrar that can quickly identify, modify, and reevaluate proposed resolutions. Implementation must incur less cost than a COTS customization, currently projected at $220 million based on the initial estimate completed by the government (LaRivee 2017).

To this end, we introduce a mixed integer programming (MIP) model ($\mathcal{S}$) that enforces *hard* constraints (i.e., limits on classroom usage, the number of sections that can be taught and how many required courses can be delayed, requirements that athletes attend practice, and logical constraints to ensure that cadets are assigned to actively-scheduled sections without overlapping time requirements) and minimizes the violation of *flexible* constraints (i.e., teaching requests, minimum and maximum section sizes, restrictions on the number of evening sections, and meeting cadet registrations). Together, these constraints consider USAFA's policies, resources, and academic preferences. Penalties elicited from leadership reflect USAFA priorities regarding deviation from the desires of individual departments and result in a scheduling and assignment process that respects institutional rules, reduces the time spent resolving registration conflicts, and sufficiently honors concerns raised by departments. The goal is for ($\mathcal{S}$) to serve as the front end to a COTS scheduling system, building timetables and section rosters that comply with both USAFA requirements and the standard weekly framework common to such software. As a tool that will be used regularly by faculty and administrators, the input ($\mathcal{S}$) requires and the output it provides must be

intuitive and straightforward. Additionally, rather than being "black-box" software, the formulation of ($\mathcal{S}$) should retain sufficient flexibility that future shifts in USAFA preferences or requirements can be easily accounted for by someone with a moderate understanding of integer programming.

The remainder of this paper is organized as follows: First, we provide a literature review on academic timetabling; we then summarize the criteria that compose the timetabling challenge at USAFA and define several of the necessary terms and parameters. Subsequently, we describe how the previous USAFA scheduling process attempts to incorporate military and athletic requirements before detailing the mixed integer program that improves on that process. We next discuss the implementation of our work, and present results. We conclude with impact and extensions.

2.3 Literature Review

Wren (1996) describes timetabling as "...the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy nearly as possible a set of desirable objectives." Each set of potential "objects" to be placed—nursing shifts (Jaumard et al. 1998), train crews (Caprara et al. 1999), athletic competitions (Easton et al. 2001)—come with equally unique requirements and constraints. While the applications of timetabling remain broad, the growth in both number and size of secondary and post-secondary educational institutions (NCES 2015) has led to an increased focus on academic applications. University schedules pose a particularly difficult challenge; with so many stakeholders (e.g., students, professors, deans), it is often the case that a model which generates a desirable schedule is far more difficult to design than one that simply finds feasible solutions (Easton et al. 2001). While new approaches (Babaei et al. 2015) have been introduced in recent years that attempt to address these issues, more than 40% of universities create large-scale schedules via manual processes which can require as many as seventeen weeks to complete (Burke et al. 2005). According to Schaerf (1999), many researchers agree that university timetabling cannot, in fact, be completely automated due

to the number of subjective factors involved. The complete process of scheduling students to courses, teachers to courses, courses to time periods, and courses to classrooms has been termed the *population and course timetabling problem* (PCTP), and Boland et al. (2008) points out that few papers attempt to tackle it entirely. Instead, they tend to focus on one or two aspects. Feng et al. (2016), for example, develop both an integer program and a genetic algorithm that schedule courses to specific times and rooms, but consider only predicted class sizes without actually assigning students. On the other hand, Cheng et al. (2003) assume a predetermined set of course offerings and attempt to create conflict-free student assignments that meet their registration requests. Researchers such as Aubin and Ferland (1989) address both course scheduling and student assignments in their models, but do so sequentially, holding either course schedules or student assignments constant at each iteration, making small changes at each step until no further improvements can be made. Hertz (1991) develops a tabu search that improves upon this iterative method, but the size of the solution space forces both Hertz and Aubin to focus on feasibility over desirability, minimizing the number of overlapping assignments rather than maximizing instructor and/or student preferences. Müller and Murray (2010) address the issue of preference ex posteriori. After scheduling courses and assigning students using a similar local-search method, they share their initial solution online and allow students to individually view and edit their registrations within limits. One approach to more directly satisfy student and faculty preferences is to assign courses and students concurrently, rather than sequentially. Bakir and Aksop (2008), along with Boland et al. (2008), take such an approach, but must reduce the number of decision variables in their models by partitioning students and courses and then scheduling these common "blocks," rather than individual students or sections. In addition to problem size, such concurrent consideration impacts the complexity of the problem. Dostert et al. (2016) prove that while students can be assigned to a specific number of sections within a given timetable in polynomial time, respecting student registrations immediately transforms the problem into an NP-complete one.

Because model complexity and associated instance size are both concerns, it is not surprising that there has been a large emphasis placed on approximation techniques and heuristics capable of generating practical university schedules in a reasonable amount of time. Lewis (2008) provides both a summary and a taxonomy of many such heuristics that have been presented and/or implemented in recent years. In addition to the integer programming and tabu search methodologies, a variety of computational techniques including simulated annealing (Tuga et al. 2007, Aycan and Ayav 2009), genetic algorithms (Khonggamnerd and Innet 2009, Alsmadi et al. 2011) and hybrid colony optimization (Fong et al. 2014) have been applied to solving different aspects and instances of university timetabling. For the most part, however, while these approaches may differ in how they *search* for a solution, they are very similar in how they *rate* a solution. Babaei et al. (2015) explains that "feasibility" of a solution is normally dictated by hard constraints which cannot be violated, regardless of preference (e.g., the number of available seats in a classroom, the fact that a student cannot be in two places at once), while "desirability" of a solution is measured by the degree to which soft constraints (e.g., no evening classes, even distribution of students among sections) are violated. Even with this structure, however, many universities remain unable to generate schedules without some form of manual manipulation. This is partially due to the fact that while sections and students can be scheduled at different times, the number of each is often fixed, meaning the only way to "improve" a given solution is to permute section start times and attempt to reassign students. In their work concerning conference seminar scheduling, Eglese and Rand (1987) introduce the flexibility to choose how many of a particular session to offer to better meet attendee interest. While their approach has obvious application to tailoring a course timetable to student registration, its practical success is heavily dependent upon the fact that the number of seminars and attendees to be scheduled is quite small, approximately 20 and 300, respectively, relative to the course offerings and class size of an average university.

Perhaps closest to our own challenge, Sampson and Weiss (1995) build on Eglase and Rand's work by developing an integer program that considers section and student preferences simultaneously while adding capacity limits. A single integer program that handles course and student scheduling would easily integrate into a COTS system, and, once developed, its parameters and penalties can be adjusted over time to reflect evolving USAFA priorities without major changes to the program's underlying structure. A practical implementation of their approach (Sampson et al. 1995), however, leaves more than 5% of student registrations unmet, requires the dean to predetermine the total number of sections, and applies a heuristic because instances of more than 1,000 students and 40 sections make solving an integer program intractable. Although the computational power available to solve integer programs has greatly grown in the past two decades, researchers like Lübbecke (2015) still hold that accommodating student sectioning while scheduling courses makes large-scale instances too complex to solve. Instead, many recent integer programming approaches to real-world university timetabling, such as Phillips et al. (2015), create models that focus solely on scheduling classes, and assume the student assignment is completed either a priori or ex posteriori.

Undergraduates at USAFA, however, represent not only students pursuing a degree, but cadets being trained for a commission; any feasible course schedule must necessarily consider both of these missions. In order for USAFA to realize the benefits of an integer program, we introduce several methods that increase the tractability of our own instance which includes 4,000 cadets and 400 courses across 1,000 sections. Furthermore, in addition to capacity constraints, graduation deadlines require that our model limit allowable registration conflicts based on future class availability. Lastly, many military duties do not begin and end according to academic semesters and require that courses and duties be deconflicted according to the "fraction" of the semester they occupy. A detailed description of specific constraints related to our integer program is provided in the appendix.

2.4 Definitions and Problem Statement

Before describing the previous USAFA model and introducing ($\mathcal{S}$), we define the necessary technical terms used throughout this paper and the requirements for a feasible and desirable USAFA schedule.

Block: A specific fraction of a semester required by a course. For example, we divide our semester into three blocks; thus, "Block 2" would refer to the time during the second one-third of the semester.

Period: A predetermined continuous span of time on a specific weekday, e.g., 0900-1030 on Monday.

Epoch: A continuous span of time during which a unique set of periods overlap. For example, given periods on Monday from 0800-0900, 0800-0915, and 0800-1000, three epochs would exist: 0800-0900, 0900-0915, and 0915-1000.

Course: A specific academic, athletic or military requirement for which cadets can register (e.g., Calculus 101, Intramural Football, or Airmanship).

Pattern: A set of blocks, with an associated set of non-overlapping periods, e.g., {{1,2}, {(Tuesday 0800-1000), (Thursday 0900-1100)}}.

Section: A $(course, pattern)$ instance to which cadets can be assigned. Courses with large numbers of registrations may require multiple sections.

Registration: A $(cadet, course)$ requirement pair.

Cadet Assignment: A designated $(cadet, section)$ pair.

Course Offering: A set of patterns for all courses that possess registrations.

Room Type: A category assigned to potential locations at which a section can meet, defined based on seating capacity and amenities, e.g., academic vs. athletic.

Registration Conflict: A registration that cannot be matched to a cadet assignment within a feasible course offering.

An acceptable USAFA scheduling process must include a set of periods which departments can use to build a course offering. This Preferred Course Offering (PCO), along with

section enrollment minima and maxima, accommodates the total number of registrations for all academic courses and represents the preferred teaching schedule of a department's instructors in aggregate. (Departments make specific section-instructor assignments once an offering is finalized.) The PCO is combined with all registrations, as well as the room type and blocks required by each course. Using this input, a model must develop a Revised Course Offering (RCO) along with a set of cadet assignments. Patterns and sections within the RCO should match those of the PCO whenever possible and must adhere to limits concerning room-type availability within a given epoch and block. Cadet assignments should obey the preferred section minima and maxima, minimize the number of registration conflicts, and must not require a cadet attend periods which overlap during a particular block.

2.5 Previous USAFA Scheduling Model

Cadets must carry an average of six courses per semester to accumulate 141 semester hours and meet their four year commission deadline. Cadets construct a four-year plan and, each semester, meet with their advisers to review and/or adjust registration for the following semester. The outcome of these meetings provides course enrollment numbers for the upcoming semester which, in turn, are used by departments to generate their PCO.

While the registrar has the ultimate responsibility of creating the upcoming semester's course offering and cadet assignments, individual departments are more keenly aware of their own staffing and course sequence requirements. To this end, creating an effective schedule has always been a lengthy and iterative dialog between departments and the registrar. Not only does each department have a dedicated representative assigned to this process, but each one is given an eight-page guideline for building its initial PCO. These rules are the outcome of years of information and best practices to try to create an initial course offering that is capable of meeting as many cadet registrations as possible. For example, courses with only a single section must use morning periods to minimize conflicts with afternoon athletics. Courses with more than ten sections must utilize all periods

evenly. The desired result of this process is a PCO that allows most registrations to be matched to feasible cadet assignments. After all possible assignments are made, remaining registration conflicts are addressed individually.

To accommodate this process, the *Previous USAFA Model* divides the semester uniformly; every school day has seven 53-minute periods, and every course meets exactly 40 times. Rather than assigning courses to specific patterns, all courses simply meet on alternating weekdays. So, in a typical, two-week period, a cadet attends a course five days; Monday, Wednesday, Friday, followed by Tuesday and Thursday. However, interruptions such as holidays may result in situations in which a course meets as few as three times in a two-week period.

This uniform scheduling structure exists largely to simplify the registrar's task of generating an RCO and resolving the subsequent registration conflicts. By not using patterns that depend on specific days and requiring that all periods be of equal length, the registrar can more easily move sections and modify cadet assignments during her process without excessive cascading effects. This rigid uniformity, however, significantly reduces the freedom departments possess to teach courses according to pedagogy and results in courses which meet either more or less frequently than is necessary. At present, a standard USAFA semester requires 18.5 weeks in order to meet the necessary contact time with cadets. The school day is also limited, starting at 0730 and ending at 1430 because, rather than including it as part of the scheduling process, it is deemed necessary to reserve a large portion of the afternoon to accommodate any possible military and/or athletic training a cadet might require.

Under the old regime, the registrar repeats the following steps until a satisfactory schedule is built:

1. Identify cadets and their registrations.

2. Solicit an initial PCO from departments.

3. Identify registrations which cannot be matched to assignments under the PCO.

4. Assign all non-athlete cadets to intramural sports and assign all athletes to their intercollegiate sport's practice requirements.

5. Find cadet assignments for as many registrations as possible using a simple greedy heuristic.

6. Use an exhaustive search to try and find cadet assignments for any remaining registrations.

7. Identify cadets with assignments that can be exchanged for another feasible assignment and determine if any such alteration permits the resolution of a lingering registration conflict.

8. Manipulate cadet assignments in an attempt to fill sections currently below the minimum preferred section size.

9. Generate a report with the algorithm's RCO and its remaining conflicts and deficiencies.

The above algorithm has three primary shortcomings: (*i*) subjective human decisions result in solutions that are not reproducible; with the exception of Steps 5 and 6, all actions reflect a *manual* effort; (*ii*) it is focused almost entirely on reducing registration conflicts and respecting section enrollment limits and largely ignores any measure of preference; and (*iii*) the greedy heuristic used by the algorithm is short-sighted; in addition to being completely undocumented and severely outdated, it considers each registration, in no particular order, and matches it to the first available assignment, also evaluated in no particular order and with no consideration of cascading effects. Not only does this approach not guarantee an optimal solution, but the registrar's guidelines and conflict resolution techniques leave

approximately 200 cadets with unassigned registrations for at least some of their required courses.

These shortcomings force the registrar and departments to invest weeks of effort, after the RCO is generated, to resolve the remaining registration conflicts either by over-filling a current section, obtaining approval for additional sections to be taught, delaying the assignment of registrations to a later semester, and/or requesting a complete or partial PCO resubmission from departments and restarting the process. This resolution must occur each semester to ensure that every cadet is able to receive their commission in four years.

2.6 Proposed Mixed-Integer Programming Approach

In order to generate an RCO and cadet assignments that result in fewer conflicts and a shorter resolution process, ($\mathcal{S}$) uses three primary decision variables: Z_{cp}, Y_{ucp}, and D_{uc}. Like the RCO generated by the registrar's algorithm, the solutions produced by ($\mathcal{S}$) do not match sections to specific rooms or instructors. Instead, Z_{cp} determines how many of section (c, p), given as a course pattern pair, to schedule, Y_{ucp} assumes a value of one if registration (u, c) of a student to a course is assigned to such a section, and the binary D_{uc} reflects whether a registration (u, c) is delayed to a later semester. To keep USAFA decision makers apprised and to simplify the problem (without compromising optimality), departments consider specific instructor allocation during the creation of the PCO and make explicit assignments ex posteriori.

Our formulation produces a solution provable to within a 5% optimality gap and provides three unique advantages over similar mixed integer programming approaches to academic timetabling (see MirHassani and Habibi (2013) for examples.): (*i*) it generates a course offering and cadet assignments *concurrently*, (*ii*) the solutions accommodate unique Academy challenges such as the period and block requirements of military training, the specialized needs of airmanship and athletics, and the necessity of on-time commissioning; and (*iii*) the problem remains tractable despite the magnitude of our effort, which includes

thousands of cadets and hundreds of courses in contrast to instances found in the literature involving hundreds of students and only dozens of courses.

2.6.1 Model Description

Before departmental preferences can be considered, ($\mathcal{S}$) must include several *hard* constraints that cannot be violated. All intercollegiate athletic practice registrations must be assigned. The total number of sections in a single epoch can never exceed the total number of rooms of a particular type. Aside from department preferences, faculty size and capabilities enforce a hard limit on how many times a specific section can be repeated in a course offering. Logical constraints require that each cadet assignment be matched to a section that exists within the associated offering, and that no cadet be required to attend two sections with overlapping patterns. Lastly, in order to achieve commissioning deadlines, each cadet may delay at most one registration per semester, and the number of delayed registrations for each course must be limited based on the future availability of that course.

With restrictions and mandates met, the remaining *flexible* constraints in ($\mathcal{S}$) promote adherence to the preferences submitted by departments. This includes using sections from the PCO wherever possible, minimizing the number of cadets assigned to evening periods, and seeking to assign cadets within section-specific minimum and maximum capacities. The penalties associated with violating these preferences are elicited from USAFA leadership and use registrations in conjunction with the faculty PCO to measure the impact of each infraction. For example, there is a preference for using only the sections requested by the PCO for a particular course; however, if additional sections must be used, leadership mandates that (i) the penalty be inversely proportional to the number of sections the PCO requests for the course, and (ii) the penalty for using a section *outside* of the PCO be double that of repeating a requested section. Although USAFA leadership initially mandates the cost of violating each flexible constraint, we conduct a sensitivity analysis (see appendix) to determine the impact of these values and recommend changes, if appropriate.

Finally, registrations which can be neither assigned nor delayed under any feasible course offering result in a registration conflict. While a solution without registration conflicts is preferred, in practice this is not possible to achieve and, thus, the constraint must remain flexible to preclude the model from being infeasible.

Registration conflicts can only be resolved through direct manual intervention, either by an adviser working with a cadet to modify their four-year graduation plan, or by USAFA leadership to approve an exemption or authorize the additional manpower required for supplemental or larger sections to be taught. To minimize the number of conflicts and the associated resources necessary to do so, each violation carries a penalty that is high relative to those associated with other flexible constraints (i.e., six times, on average) and proportional to the credit-hours associated with the course responsible for the conflict.

2.6.2 Tractability

The structure associated with our model and the number of variables and constraints in realistic instances puts significant strain on current hardware, software and MIP solution techniques. The computational complexity of a constrained decision problem such as ($\mathcal{S}$) falls in the category of nondeterministic polynomial time complete (NP-complete) problems, signifying that the problem is among the hardest to solve because there is no known polynomial-time solution algorithm. Moreover, the average USAFA enrollment includes approximately 4,000 cadets, each of whom possesses up to eight registrations per semester. For instances that contain 400 courses and 100 different periods, ($\mathcal{S}$) can consist of billions of variables, and nearly twice as many constraints, and can require days to find a feasible solution that adheres to the proposed PCO. In practice, we mitigate this via several techniques designed to increase problem tractability: (*i*) eliminate unnecessary variables and constraints through set indexing and by combining certain penalties; (*ii*) add strengthening constraints; (*iii*) tune the solver software; and (*iv*) introduce a procedure to obtain an initial feasible solution. The appendix provides specific details for the application of each technique.

2.7 New Scheduling Process

In order to facilitate the transition from the previous USAFA model to implementation of ($\mathcal{S}$), we present a simple, four-step iterative procedure that closely mirrors the previous process, but gives departments more options without having them involved in behind-the-scenes optimization mechanics.

2.7.1 Request for PCO

During a semester, after cadets have met with their advisers and registered for courses, an initial PCO for each department can be formed. To this end, we present each department with the new available time periods (Figure 2.1).

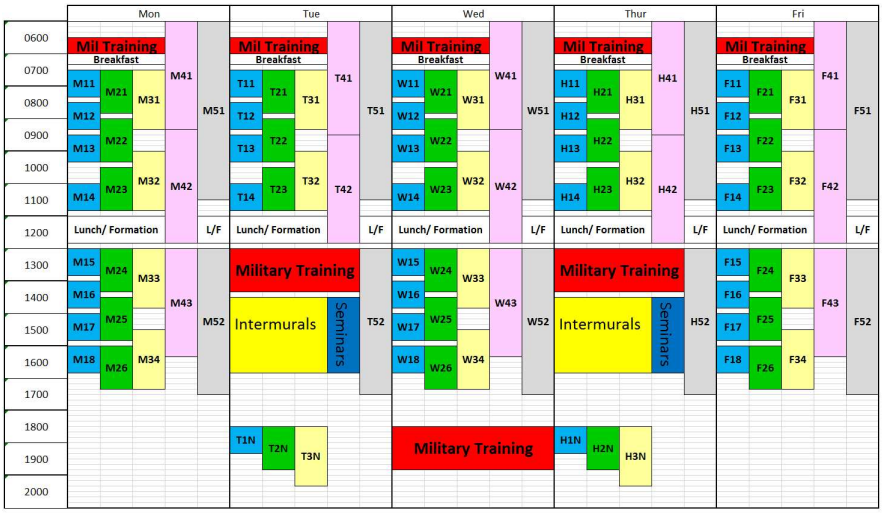


Figure 2.1: Matrix of available weekly time periods: In comparison to the previous USAFA Model, departments are allowed eight additional times at which periods can start and six different period lengths. These may be combined into any desired weekly pattern.

Because ($\mathcal{S}$) ensures military training and athletics requirements are met, departments are encouraged to build their PCO in a manner they deem pedagogically appropriate. Rather than attempting to adhere to the afore-mentioned complicated guidelines, ($\mathcal{S}$) determines the feasibility of meeting their preferences, and provides alternatives when necessary.

2.7.2 PCO Submission

A PCO is submitted from each department in the form of a spreadsheet that contains four pieces of information for each unique section a department plans to offer: (*i*) the associated course; (*ii*) a set of periods from Figure 2.1; (*iii*) the number of times a department would like to offer the section; and (*iv*) a minimum and maximum number of cadets preferred in each section. Table 2.1 provides an example.

Table 2.1: PCO Input: Sample of departmental preferences for course periods, patterns and sizes. While not reflected in this example, patterns with more than three periods are allowed.

Course Number	**1st Period**	**2nd Period**	**3rd Period**	**Number of Sections**	**Minimum Enrollment**	**Maximum Enrollment**
c1523	M12	W12	F12	3	10	20
c2317	F41	-	-	1	0	50
c8921	T26	H26	-	5	15	25
...	...	...	...	...	...	...

Registrations and additional course information from the registrar are added to the PCO data. This includes which courses have special location requirements (e.g., airfield or physical education), which can be delayed, and which require the entire semester versus only certain blocks.

2.7.3 Data Additions and Processing

To facilitate the implementation of ($\mathcal{S}$), preprocessing code forms the PCO and registration data into sets which it then expands upon. This includes simple "rearrangements" of data for constraint qualifications (e.g., constructing $\mathcal{U}_c$, undergrads registered for course c, from the registrar's $\mathcal{C}_u$, courses for which undergrad u has registered) or more convenient groupings (e.g., collecting all patterns for a particular course c into a single set $\mathcal{P}_c$). More importantly, in addition to rearranging the given data, code facilitates the following:

First, using the preferences from the PCO as a template, additional sections are included as alternatives to those requested by the PCO. For example, in Table 2.1, while course c2317 is preferred to be offered only as a pattern involving the single period on Friday (F41) by the *department*, this does not consider possible *registrar* complications (e.g., a cadet with a registration for another class whose only sections involve overlapping patterns). With this in mind, the code relaxes ($\mathcal{S}$) to consider additional sections that mimic the patterns requested in the PCO, each of which is assigned capacity parameters (i.e., enrollment minima and maxima) identical to those in the PCO, with the exception that their preferred number of sections is set to zero. In our example, twelve additional single-period sections of course c2317 are made available using the periods from Figure 2.1 that match the length of (F41). The model ($\mathcal{S}$) now has more feasible sections in which to schedule c2317, but is penalized for choosing any pattern outside the department's PCO. Second, the code provides the option of generating an initial solution, which assigns seniors, juniors and sophomores successively, fixing the assignments of each class before proceeding. By partitioning the problem, the number of variables is controlled at each step to provide a quick, feasible solution that prioritizes upperclass requirements and which ($\mathcal{S}$) can use as a starting point.

2.7.4 Output and Feedback

Once ($\mathcal{S}$) solves, two files are generated and returned to departments: (*i*) summary statistics of the solution which highlight both the deviation from the original PCO and the number of registration conflicts remaining (Table 2.2); and (*ii*) interactive course and cadet schedules under the current solution (Figures 2.2 and 2.3).

Figure 2.2 provides a measure of how closely the proposed solution matches the PCO, allowing the registrar and departments to gauge the extent to which the initial PCO meets cadet requirements. The course schedule in Figure 2.2 serves as a tool with which departments can evaluate the proposed solution and decide whether or not the changes represented are feasible based on staffing and course material.

Table 2.2: PCO Deviation: Numbers represent the difference between the PCO request and solution proposed by ($\mathcal{S}$). Negative numbers signify PCO requests which were unused, positive values indicate sections added by ($\mathcal{S}$), and "-" corresponds to perfect agreement between the PCO and proposed schedule.

Summary Statistics:
Registration Conflicts: 24
Total PCO Sections Requested: 1,451
Unmet PCO Requests: 325
Non-PCO Sections Used: 404

Blocks	Course	**Total deviation from PCO by period and block**						
		M11	M12	M13	M14	M15	M16	M17
1-3	c201	-	(-1)	1	3	(-2)	-	-
1-2	c218	-	-	-	-	-	(-2)	4
1-3	c223	-	1	-	1	-	-	-
1-3	c233	1	1	-	-	-	(-1)	-
1-3	c234	1	1	(-1)	1	-	-	-
2-3	c323	-	-	-	-	1	-	-
1-3	c335	(-2)	-	-	1	1	1	1

The cadet schedule in Figure 2.3 is intended to aid the registrar in focusing on those cadets with registration conflicts and evaluating the work required to resolve them. Once these evaluations have been made, the proposed solution can either be accepted as-is or with one-off modifications. Individual departments can resubmit a modified PCO based on the registration conflicts and unmet requests associated with their original preference. The process is repeated until it produces a solution that is acceptable to both the registrar and the departments.

2.8 Results

The Superintendent has contracted with a third-party vendor to work with the registrar and Information Technology department at USAFA to develop the front-end which integrates ($\mathcal{S}$) into COTS scheduling software and facilitates information sharing and processing between departments and the registrar. In order to reflect the hardware available

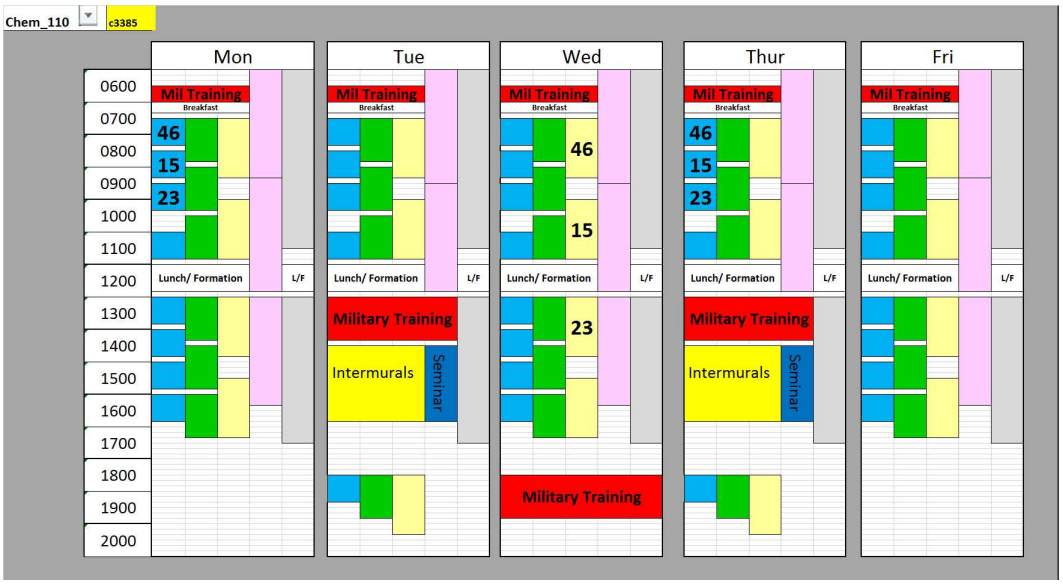


Figure 2.2: Interactive Course Schedule: Enrollment numbers by period for an arbitrary course (Chemistry 110)

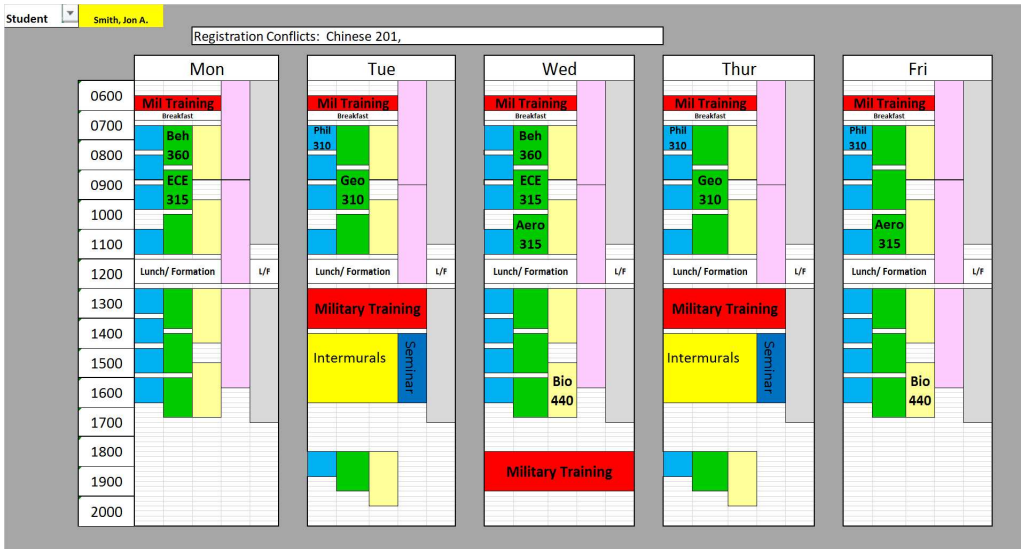


Figure 2.3: Interactive Cadet Schedule: Weekly schedule for an arbitrary cadet including a list of registration conflicts.

to the registrar, we use 32 Intel(R) Xeon(R) 2.60GHz processors with 32 GB of RAM running CPLEX 12.6.2 to obtain our results. Decision regarding the implementation of ($\mathcal{S}$) was based on USAFA registration data for the 2016 Fall semester. A request for a PCO with the newly structured periods (Figure 2.1), along with appropriate instructions, was sent to all departments. The returned PCO includes 1,401 desired sections, 1,069 of which were unique. The registrar provided 30,986 registrations from 3,894 cadets for 459 courses (approximately eight registrations per cadet).

While the process of resolving all conflicts and publishing a finalized schedule requires multiple iterations between departments and the registrar, the time required for that process is largely defined by the quality of the initial attempt to merge PCO and registration requirements into a revised course offering (RCO) and set of cadet assignments. The quality of this solution is measured using four separate indicators:

Registration Conflicts: Fewer conflicts indicate more registrations have been met and, thus, the solution requires that the registrar spend less time arbitrating resolutions with leadership and departments.

Unmet Preferences: The number of sections from the PCO that were not used by the RCO; a smaller value indicates closer adherence to departmental preferences and a solution which they will more readily accept.

Total RCO Sections: The total number of sections required by the RCO; smaller values represent less strain on faculty resources and a solution USAFA leadership will more readily approve. Further, this measures how many alternative, non-preferred sections were needed in lieu of any Unmet Preferences.

Time: Hours required to generate the RCO; shorter run times provide quicker feedback to departments concerning the feasibility of their PCO and an estimate of the time required to evaluate subsequent PCO revisions.

Table 2.3 compares the initial RCO generated by four methods: (*i*) the *Previous USAFA Model*, an almost completely manual effort that uses the alternating-day format with fixed 53-minute periods; (*ii*) a *Basic MIP* based on an initial proposed formulation (Gonzalez 2011) and developed by contractors, it contains only fundamental USAFA constraints that focus on feasibility over desirability and lacks our tractability enhancements; (*iii*) our proposed *($\mathcal{S}$) Formulation*, described in the previous section, which integrates faculty preferences along with tractability modifications in order to generate schedules that are viable *and* desirable in a reasonable amount of time; and (*iv*) an implementation of ($\mathcal{S}$) using the initial feasible solution obtained from the algorithm presented in the appendix. We solve all models to a 5% optimality gap. This gap was chosen by USAFA because, in practice, it was found that using a smaller gap greatly increases solution times without significant improvement in solution quality.

Table 2.3: Results: Statistics for the Fall 2016 semester (solved to within 5% of optimality, where applicable); solution times represent a single iteration. †The Previous Model is primarily a manual effort; the time listed accounts only for the greedy algorithm and omits time spent during manual manipulation. ††The Basic MIP is unable to reach a solution using the initial hardware; the parenthetical numbers reflect results obtained by increasing system memory. †††Variable and constraint counts pertain to ($\mathcal{S}$) using the improved solution provided; time includes that spent finding an initial solution (See appendix).

	Instance Characteristics		Solution Quality			
	Number of Variables	**Number of Constraints**	**Registration Conflicts**	**Unmet Preferences**	**Total RCO Sections**	**Time (Hours)**
Previous USAFA Model†	NA	NA	200	0	1,682	0.5
Basic MIP††	349,932	219,353	(41)	(563)	(1,280)	(106)
($\mathcal{S}$) *Formulation*	221,180	99,573	44	147	1,359	1.6
($\mathcal{S}$) + Initial Solution†††	222,609	99,224	43	74	1,430	1.3

The *Previous USAFA Model* generates a schedule which uses only PCO sections because it is not equipped to investigate alternatives. This comes at the price of 200 registration conflicts which require a sizable effort on the part of the registrar to resolve; each adjustment

must be made manually and can cause cascading conflicts. After generating the RCO reflected in Table 2.3, the *Previous USAFA Model* requires that the registrar work with departments to make an average of 2,715 adjustments per day affecting more than 1,000 cadets. The *Basic MIP* in Table 2.3 closes the optimality gap to only 23% before exhausting system memory. Run on a machine with increased memory, it reaches the desired optimality in just over 106 hours. The solution greatly reduces registration conflicts to 41, but ignores over 40% of the section preferences from the PCO. This solution requires fewer fixes to cadet assignments at the expense of teaching preferences, indicating that departments would likely resubmit a revised PCO and wait four additional days for it to be evaluated. On the other hand, ($\mathcal{S}$) achieves a similar registration conflict count of 45, and does so in 1.6 hours; additionally, it respects 90% of department preferences. Although ($\mathcal{S}$) uses 79 more sections than the *Basic MIP*, the total number is far below that used by the *Previous USAFA Model* and remains less than the 1,401 requested in the PCO. Figure 2.4 shows that it is possible to reduce the total number of sections, but only at the expense of additional unmet preferences. Prior to implementing ($\mathcal{S}$) into the new COTS software, the registrar set a benchmark of 60 registration conflicts, satisfied that this goal significantly reduces the required man-hours to finalize a schedule. Solutions found by ($\mathcal{S}$) surpass this goal by 25%, and their close adherence to preferences within the PCO has led to an overwhelming recommendation from the USAFA Faculty Senate to adopt the new schedule of calls.

Using the initial solution provided by the algorithm in the appendix, ($\mathcal{S}$) generates a solution in 19% less time (including time spent on the algorithm); the schedule incurs approximately half of the unmet preferences as ($\mathcal{S}$) does without the initial solution, but uses 71 additional sections. The difference in quality metrics is consistent with the tradeoffs illustrated in Figure 2.4, and can be controlled in subsequent iterations if the initial RCO and cadet assignments are not accepted.

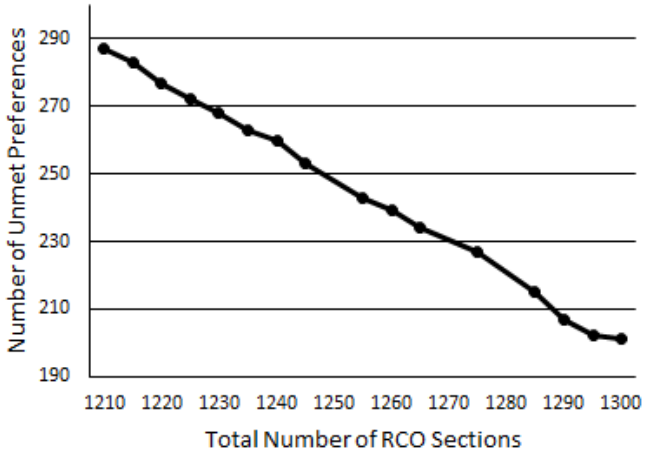


Figure 2.4: Number of unmet PCO preferences as a function of restricting total number of sections offered; it is possible to reduce the total number of sections required by the RCO, but only at the expense of additional PCO deviation.

2.9 Impacts and Future Work

Scheduling courses and cadets via ($\mathcal{S}$) represents key advantages over the *Previous USAFA Model* in flexibility of use, quality of solution, and execution time. With the diversity of options provided in Figure 2.1, many instructors report that using a twice-a-week 80 minute pattern (not available under the *Previous USAFA Model*) greatly improves their 'effective' contact time, while those who wish to retain their current lesson plans continue to do so using thrice-a-week 50 minute patterns. Replacing the previous alternating-day approach with a repeated-week pattern allows ($\mathcal{S}$) to accommodate both pattern preferences and generate a schedule in which the required contact-hours for a semester are met in 15 weeks, 3.5 fewer than the *Previous USAFA Model*. Furthermore, with a total proposed cost of $100 million, incorporating ($\mathcal{S}$) into a COTS system represents $120 million in savings in comparison to maintaining the previous alternating-day model (LaRivee 2017). Satisfied that the analysis and successful application of ($\mathcal{S}$) meets or exceeds the conditions imposed on the Pathways Committee, the Superintendent has approved the use of ($\mathcal{S}$) and has made the decision to obtain a Request for Proposal from industry to implement COTS

software and replace the current Oracle system. This decision fundamentally changes the way in which the Academy has approached its schedule of calls for more than 50 years, and has already seen benefits. During early implementation, an economics course was able to combine two separate lessons into a single night class. The new structure eliminates time spent re-summarizing material and allows professors to "... get twice as much done in less than two full periods of time" (Branum 2014). By shortening the overall semester, ($\mathcal{S}$) is also opening new opportunities for military field training.

During the writing of this paper, the Pathways Committee modified the matrix in Figure 2.1 several times in response to requests from USAFA leadership. In each of these cases, the core design of ($\mathcal{S}$) required no changes. Instead, only minor set and parameter updates were necessary to implement the revisions. The flexibility in the application of ($\mathcal{S}$) not only allows it to evolve with USAFA's needs, it introduces additional applications within the university. Work has begun to use ($\mathcal{S}$) to plan cadet flight schedules around their academic ones. Building these two types of schedules in concert maximizes flight time during good weather and reduces potential safety issues. Further, ($\mathcal{S}$) has also proven useful as a "what-if" decision tool outside of the scheduling process to determine cost-benefit analysis of hiring additional professors or permitting more sections during evening or weekend hours in future semesters.

In addition to expanding ($\mathcal{S}$)'s applicability, heuristics can reduce its run time. Tests show that our own process for finding an initial solution may help achieve this goal by using a different prioritization scheme. Although class year is an intuitive option, other partitioning choices such as athletics or degree path may prove more beneficial.

Implementing new procedures in any organization presents challenges; it requires co-operation from a broad range of stakeholders. USAFA is no exception, with an annual budget in the hundreds of millions and a workforce in the thousands, overhauling a half century of dogma on how cadets should spend their time is no simple task. However, most agencies on campus have finally agreed that "... the schedule of calls, as it currently exists,

is an obstacle, ... cadets are suffering" (Branum 2014). The flexibility offered by ($\mathcal{S}$) is the first step in refocusing on cadets and ensuring that USAFA remains competitive among four-year universities while accomplishing its military mission.

2.10 Formulation ($\mathcal{S}$)

Our formulation of ($\mathcal{S}$) does not include an explicit objective function. Instead, we use notation borrowed from Brown et al. (1997) in which the goal of the program is simply to minimize the amount by which elastic constraints (denoted by $\doteq$ or $\dot{\leq}$) are violated.

Sets:

$\mathcal{U}$: Undergraduate cadets

$\mathcal{C}$: Courses

$\mathcal{T}$: Time periods

$\mathcal{P}$: Patterns

$\mathcal{B}$: Blocks

$\mathcal{E}$: Epochs

$\mathcal{R}$: Classroom types

$\mathcal{T}^V \subset \mathcal{T}$: Evening time periods

$\mathcal{C}^D \subset \mathcal{C}$: Courses which can be delayed to a later semester

$\mathcal{C}^I \subset \mathcal{C}$: Intercollegiate athletics courses

Indexed Sets:

$\mathcal{C}_b \subset \mathcal{C}$: Courses which require block b

$\mathcal{C}_r \subset \mathcal{C}$: Courses which require room type r

$\mathcal{U}_c$: Cadets registered for course c

$\mathcal{C}_u$: Courses registered for by cadet u

$\mathcal{P}_c$: Available patterns for course c

$\mathcal{C}_p$: Courses which are compatible with pattern p

$\mathcal{M}_p$: Time periods included in pattern p

$\mathcal{F}_p$: Time periods which overlap those in $\mathcal{M}_p$

$\mathcal{T}_e$: Time periods which overlap epoch e

Parameters:

σ_{cp} = Number of preferred (c, p) sections (from PCO)

$\overline{\sigma}_c$ = Maximum number of sections of course c that can be scheduled in a semester

$\underline{g}_{cp}$ = Minimum preferred number of cadets assigned to section (c, p)

$\overline{g}_{cp}$ = Maximum preferred number of cadets assigned to section (c, p)

$\overline{d}_c$ = Maximum number of cadets allowed to delay course c

$\overline{s}_r$ = Maximum number of rooms of type r available

M_{cp} = "Big-M" bounding the number of possible cadet assignments to section (c, p)

M_u = "Big-M" bounding the number cadet assignments possible for cadet u

Variables:

Z_{cp} = Number of (c, p) sections scheduled

Y_{ucp} = 1 if a cadet u is assigned to attend course c using pattern p, 0 otherwise

D_{uc} = 1 if cadet u delays course c, 0 otherwise

Constraints:

$$Z_{cp} \dot{\leq} \sigma_{cp} \qquad \forall c \in \mathcal{C},\ p \in \mathcal{P}_c \tag{2.1}$$

$$\sum_{p \in \mathcal{P}_c} Z_{cp} \leq \overline{\sigma}_c \qquad \forall c \in \mathcal{C} \tag{2.2}$$

$$\sum_{c \in \mathcal{C}_r} \sum_{p \in \mathcal{P}_c:\ \mathcal{M}_p \cap \mathcal{T}_e \neq \emptyset} Z_{cp} \leq \overline{s}_r \qquad \forall r \in \mathcal{R},\ e \in \mathcal{E} \tag{2.3}$$

$$\sum_{u \in \mathcal{U}_c} Y_{ucp} \dot{\leq} \overline{g}_{cp} Z_{cp} \qquad \forall c \in \mathcal{C},\ p \in \mathcal{P}_c \tag{2.4}$$

$$\sum_{c\in\mathcal{C}_p}\sum_{u\in\mathcal{U}_c} Y_{ucp} \doteq 0 \qquad \forall p\in\mathcal{P}:\ \mathcal{T}^V\cap\mathcal{M}_p\neq\emptyset \tag{2.5}$$

$$\sum_{u\in\mathcal{U}_c} Y_{ucp} \ \leq M_{cp}Z_{cp} \qquad \forall c\in\mathcal{C},\ p\in\mathcal{P}_c \tag{2.6}$$

$$\sum_{p\in\mathcal{P}_c} Y_{ucp} \ = 1 \qquad \forall c\in\mathcal{C}^I,\ u\in\mathcal{U}_c \tag{2.7}$$

$$D_{uc}+\sum_{p\in\mathcal{P}_c} Y_{ucp} \doteq 1 \qquad \forall u\in\mathcal{U},\ c\in\mathcal{C}_u \tag{2.8}$$

$$\sum_{c\in\mathcal{C}^D\cap\mathcal{C}_u} D_{uc} \ \leq 1 \qquad \forall u\in\mathcal{U} \tag{2.9}$$

$$\sum_{u\in\mathcal{U}_c} D_{uc} \ \leq \overline{d}_c \qquad \forall c\in\mathcal{C}^D \tag{2.10}$$

$$\sum_{c\in(\mathcal{C}_b\cap\mathcal{C}_u)}\left(\sum_{p\in\mathcal{P}_c:t\in\mathcal{F}_p} Y_{ucp}+M_u\sum_{p\in\mathcal{P}_c:t\in\mathcal{M}_p} Y_{ucp}\right) \leq M_u \qquad \forall b\in\mathcal{B},\ u\in\mathcal{U},\ t\in\mathcal{T} \tag{2.11}$$

$$\begin{aligned} Z_{cp} &\in\mathbb{Z} && \forall c\in\mathcal{C}\ p\in\mathcal{P}_c \\ Y_{ucp} &\in\{0,1\} && \forall u\in\mathcal{U}, c\in\mathcal{C}_u, p\in\mathcal{P}_c \\ D_{uc} &\in\{0,1\} && \forall u\in\mathcal{U}, c\in\mathcal{C}^D\cap\mathcal{C}_u \end{aligned}$$

In the absence of an explicit objective function, the constraints which define $(\mathcal{S})$ either ensure the solution is feasible under USAFA requirements or that said solution is as desirable as possible to departments and cadets. Constraint (2.1) encourages adherence to the PCO and its violation, i.e., scheduling more sections in a particular pattern than requested by the PCO, incurs a penalty based on the total number of sections initially requested. (Larger penalties are assigned to courses with fewer sections.) Constraints (2.2) and (2.3) represent strict manpower and resource constraints on the number of sections and cannot

be violated. Specifically, Constraint (2.2) limits the number of sections of a particular course can be taught per semester per departmental guidelines. Constraint (2.3) reflects physical resources by limiting all sections of all courses scheduled during a particular epoch to the number of rooms that are eligible and available across campus. Constraints (2.4)-(2.7) similarly control how cadets are assigned to course sections. Constraints (2.4) and (2.5) attempt to honor PCO requests with flexible limits on section enrollment and minimal assignment of cadets to evening sections, respectively, via penalties similar to those associated with Constraint (2.1). Constraints (2.6) and (2.7) are hard constraints that enforce a logical restriction against assigning cadets to non-scheduled sections, and that require intercollegiate athletes to attend the necessary practice (offered only in a single pattern) respectively.

Ultimately, every registration must either be met or delayed to a later semester when the course is re-offered. Historically, finding a schedule which meets these requirements and does not deviate from the initial PCO is highly unlikely. Instead, the *Previous USAFA Model* uses guidelines and heuristics to try and meet as many course registrations as it can, and then begins a lengthy negotiation process with departments in an attempt to fix the remaining registration conflicts. While ($\mathcal{S}$) has the flexibility to deviate from the PCO and generate a schedule with no registration conflicts, it does so at the cost of violating flexible constraints. Completely eliminating registration conflicts inevitably results in a schedule that is unlikely to be endorsed by all departments. Thus, violation of Constraint (2.8) remains flexible, with each violation translating into a registration conflict which must be resolved by USAFA in some fashion. Because these violations represent a measure of the man-hours necessary to adjudicate a final schedule, their penalty is high relative to those associated other flexible constraints (i.e., six times, on average) and is proportional to the credit-hours associated with the course c that is responsible for the conflict. Furthermore, while registration conflicts are definitely undesirable, delaying registrations only simplifies the current semester by complicating a later one. To this end, Constraint (2.9) enforces

a USAFA restriction that each cadet delay no more than a single registration in a given semester and Constraint (2.10) limits the number of registrations for course c that may be delayed. (The parameter $\bar{d}_c$ is based on the future availability of course c.)

Lastly, Constraint (2.11) ensures that a cadet cannot be in more than one place at a time and must account for all sections to which a cadet is assigned, the periods during which they meet, and all possible ways in which these periods may overlap within individual blocks. Initially, this requirement was modeled as two separate constraints, an equality and an inequality, in an attempt to generate additional cuts. This construction was ultimately discarded as any gains were overshadowed by the increase to the required number of variables and constraints.

The objective function is implicit in this formulation, and consists of a sum of the following terms representing weighted deviations associated with: (i) scheduling a particular section more often than is requested by the PCO: Constraint(2.1), (ii) exceeding a maximum class size: Constraint (2.4), (iii) assigning cadets to evening periods: Constraint (2.5), and (iv) incurring registration conflicts: Constraint (2.8).

2.11 Size Reduction and Bound Tightening

The following tactics reduce the size of our monolith, tighten its bounds, and provide our instances with an initial feasible solution, thus expediting solutions (Klotz and Newman 2013). We reduce the number of assignment variables by using the indexed sets (see previous section). For example, without careful control, Y_{ucp} can include approximately 10^9 binary variables. However, by ignoring instances in which a cadet u does not require a course c, as well as those in which c is not offered in pattern p, this number is drastically reduced to just over 200,000. Controlling Y_{ucp} in this way also allows constraints involving the eliminated assignment possibility to be removed, and reduces the number of constraints defined in (2.5) - (2.8). Similar reductions apply to Z_{cp}, D_{uc}, and their associated constraints.

The size of the model is further limited by removing variables and constraints associated with penalties below a certain threshold, and incorporating them elsewhere in the

model. For example, Constraint (2.4) originally included the lower bound $\underline{g}_{cp}$ on the average number of cadets assigned to each section. However, the difficulties associated with exceeding the upper bound (e.g., faculty size, room types, office hours) heavily outweigh the inconvenience of falling below the minimum. We therefore remove the constraints related to under assignment, and instead modify the penalties associated with Constraint (2.1) to be proportional to $\underline{g}_{cp}$ (i.e., as $\underline{g}_{cp}$ increases, so does the penalty for opening an additional section). This modification results 16,000 fewer variables and constraints, and eschews solving conflicts simply by opening many small sections.

While some variables cannot be explicitly removed from ($\mathcal{S}$), they can be fixed for a particular instance based on USAFA mandates. For example, in the Fall 2016 PCO, USAFA identified several specific courses which cadets must delay to a summer semester, if possible. (Resources for the summer section had already been committed.) In addition to fixing many Y_{ucp} variables a priori, any constraint which involves Y_{ucp} is also strengthened with the removal of penalty variables.

The model is also tightened via the following means: (*i*) an upper bound on the sum of assignment binaries is equal to the total number of registered courses; (*ii*) the magnitude by which flexible constraints can be violated is limited. At present, this upper bound is prescribed by USAFA leadership and is applied a priori across all sections and cadets. Future iterations of ($\mathcal{S}$) will allow input from departments to tailor individual limits to their particular needs; and (*iii*) flexible constraints associated with penalties above a certain threshold are converted into hard constraints. An example of this latter methodology involves Constraint (2.3); while not impossible, appropriating additional rooms for regular academic use is so difficult that violating (2.3) is never optimal under any solution found by ($\mathcal{S}$). As a hard constraint, (2.3) requires fewer variables, is stronger, and, as implemented now, reduces the time ($\mathcal{S}$) spends searching through feasible solutions which are never optimal in practice. The relevant node logs reveal that, in solving ($\mathcal{S}$), CPLEX never branches off the root node; all optimization is done through a combination of cuts and

internal heuristics. While our own cuts are effective, it should be possible to examine the post-processed model to identify the most powerful cuts and apply them a priori.

We also tune our solver; probing fixes binary variables involved in packing and partitioning constraints. Setting a probe value of 2 reduces our solve time in practice. Furthermore, since our variables greatly outnumber the constraints, we employ options to ensure that our solver uses a primal versus dual simplex at the root node. Finally, we introduce a heuristic to produce an initial feasible solution. All tractability enhancements described in this section are also applied to the process of finding an initial solution described below.

2.12 Initial Solution for $(\mathcal{S})$

To find an initial solution, $(\mathcal{S})$ partitions the data it receives according to year-group (i.e., sophomores, juniors, seniors) in order to solve smaller subproblems that schedule courses with unique year-group registrations. Figure 2.5 demonstrates how the derived solution affects the overall solve time and the rate at which $(\mathcal{S})$ closes its optimality gap. The pseudocode that follows (Figure 2.6) details how the initial solution is derived.

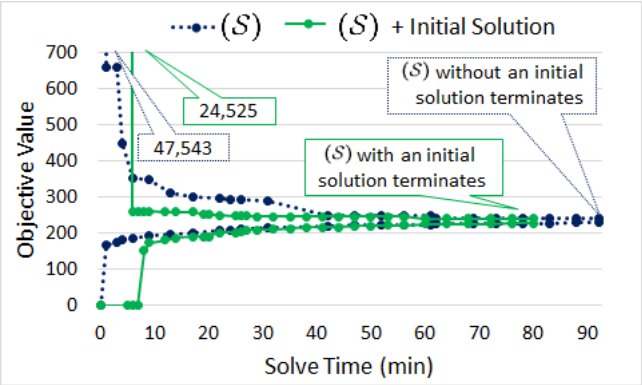


Figure 2.5: Performance of $(\mathcal{S})$ with and without an initial solution: Early on, $(\mathcal{S})$ closes its optimality gap more quickly without an initial solution; however, time spent deriving the initial solution is quickly recuperated and allows a 19% reduction in overall solve time.

2.13 Penalty Sensitivity Analysis

The formulation in the previous section includes four flexible constraints: (2.1), (2.4), (2.5) and (2.8), the violation of which are controlled by three types of penalties. (Constraints (2.4) and (2.5) are both related to overfilling a section and, thus, use similar penalties.) Although USAFA leadership provides course- and pattern-specific penalties for the violation of each constraint, we conduct our own analysis to determine the appropriateness of the magnitude of the penalties. Using increments of 25%, the penalty for each type of violation (i.e., using additional sections, overfilling a section, and incurring a registration conflict) varies from 50% to 150% of the prescribed USAFA value. Using the Fall 2016 PCO and registration data, our analysis solves ($\mathcal{S}$) for each possible combination. Due to the large number of requisite runs, we use an optimality gap of 8% to reduce the time necessary to test all 125 combinations. Figure 2.7 summarizes some of the main comparative results of the analysis.

Solutions are evaluated using the four metrics presented in our results (i.e., registration conflicts, unmet preferences, total sections, and solve time). Because registration conflicts are assigned a high penalty relative to those associated with other violations, their occurrence varies only slightly within our analysis, between 44 and 48, and is not included in the summaries provided in Figure 2.7. Concerning the remaining metrics, unmet preferences range from 96 to 222, total sections from 1,322 to 1,449, and run times from 0.5 to 1.9 hours.

The analysis shows that variation of the penalties associated with overfilling and adding sections has minimal effect on both the number of unmet preferences and the total number of sections. Furthermore, adjustments to registration conflict penalties result largely in simple tradeoffs between unmet preferences and total number of sections, as predicted by Figure 4. Subsequently, based on the results illustrated in Figure 7(a), the penalties associated with registration conflicts are increased to 125% of their original values.

Initial Solution for $(\mathcal{S})$;
`// Partition Cadets (Freshmen schedules are preset and not included)`
$\mathcal{U}_1 \leftarrow$ Senior cadets $u \in U$;
$\mathcal{U}_2 \leftarrow$ Junior cadets $u \in U$;
$\mathcal{U}_3 \leftarrow$ Sophomore cadets $u \in U$;
`// Identify courses unique to each year-group`
for $n = 1, ..., 3$ **do**
  $\mathscr{Y}_n \leftarrow \left(\bigcup_{u \in U_n} C_u \right) \setminus \left(\bigcup_{u' \in U_{n'}, n' \neq n} C_{u'} \right)$
end
`// Make cadet assignments for all non-unique courses`
$(\mathcal{S})^{\mathscr{Y}_0} \leftarrow (\mathcal{S})$ without constraints or registrations related to courses $c \in \bigcup_{n=1}^{3} \mathscr{Y}_n$;
Solve $(\mathcal{S})^{\mathscr{Y}_0}$;
`// Save assignments`
foreach Y_{ucp} **in** $(\mathcal{S})^{\mathscr{Y}_0}$ **do**
  $y^0_{ucp} \leftarrow Y_{ucp}$
end
`// Assign each year-group individually;`
`// Retain assignments for non-unique courses`
for $n = 1, ..., 3$ **do**
  $(\mathcal{S})^{\mathscr{Y}_n} \leftarrow (\mathcal{S})$ with only $u \in U_n$ and the associated registrations and constraints;
  foreach y^0_{ucp} **such that** $u \in U_n$ **do**
    Add constraint $Y_{ucp} = y^0_{ucp}$ to $(\mathcal{S})^{\mathscr{Y}_n}$
  end
  Solve$(\mathcal{S})^{\mathscr{Y}_n}$;
  foreach Y_{ucp} **in** $(\mathcal{S})^{\mathscr{Y}_n}$ **do**
    $y^n_{ucp} = Y_{ucp}$
  end
end
`// Derive initial solution using assignments from each class`
$(\mathcal{S})^I \leftarrow (\mathcal{S})$;
for $n = 1, ..., 3$ **do**
  Add constraint $Y_{ucp} = y^n_{ucp}$ to $(\mathcal{S})^I$
end
Solve $(\mathcal{S})^I$;
Return solution;

Figure 2.6: Pseudocode to derive an initial solution for $(\mathcal{S})$.

Beyond reducing average runtimes by approximately 20%, this increase strengthens the emphasis placed on avoiding registration conflicts. The eight additional unmet preferences this change incurs represent a change of only 3.8% and, as previously stated, are offset by the eight fewer total sections it requires.

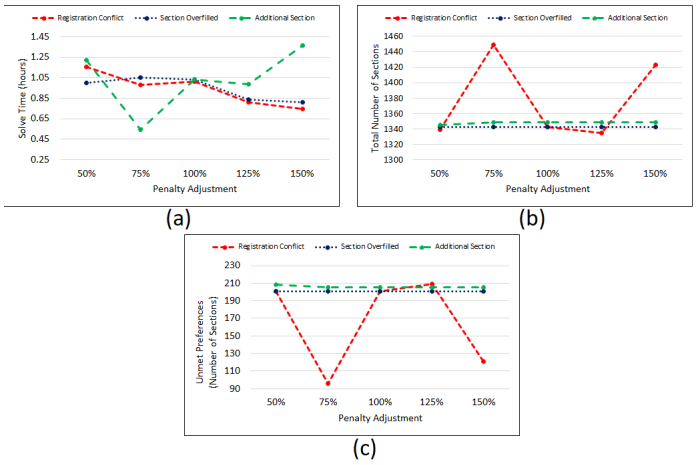


Figure 2.7: Quality metrics (a) Solve time; (b) Total sections; and (c) Unmet preferences as functions of adjustments to the penalties for flexible constraint violation. For each violation, i.e., registration conflict, overfilling a section, and using an additional section, the associated dashed line represents the result of adjusting its penalty while all others are held constant. The trend lines in each chart are slightly perturbed for ease of visualization.

CHAPTER 3

MINIMUM-RISK ROUTING THROUGH A MAPPED MINEFIELD

A paper submitted to the journal *Networks*.

Christopher Richards [6] [7], Chris Odom [8] [9], David Morton [10], Alexandra Newman [11]

3.1 Abstract

We embed a directed graph $G(V, E)$ in a representation of a naval minefield; vertices V represent waypoints and edges E denote possible segments for ship transit. A new model identifies a simple s-t path through the minefield that minimizes the risk of incurring unacceptable damage from threats, i.e., mine detonations. Traditional "edge-additive" models rely on shortest-path algorithms that over-accumulate measured risk along a path. Our "threat-additive" approach accumulates risk based upon the path's closest point of approach to each mine. We formulate and solve this model as both (i) an integer program and its stronger variant, and (ii) via an A* search algorithm, coded in C++. A preprocessing heuristic reduces runtimes. In the case of (ii), an additional heuristic based on Voronoi diagrams often improves this initial solution. In realistic instances of G ($|V| = 963$, $|E| = 2,793$), the traditional model solves in less than 0.1 second, but returns sub-optimal solutions in 43% of instances. By contrast, the integer programs find solutions in fewer than 90 seconds, on average, but produce objective function values that yield an average of 7% less risk. Using Voronoi graphs, the A* search returns solutions with 15% less risk than the traditional model in one second, on average.

[6]Major, United States Air Force.

[7]PhD Candidate, Mechanical Engineering Department, Colorado School of Mines.

[8]Lieutenant, United States Navy

[9]Shift Engineer, Nuclear Power Training Unit, Charleston, SC

[10]Professor, Department of Industrial Engineering and Management Sciences, Northwestern University

[11]Professor, Department of Mechanical Engineering, Colorado School of Mines.

3.2 Introduction

With dwindling support for traditional Mine Countermeasure Vessels (MCMVs) (McNally 2012), the U.S. Navy requires a way to minimize the risk of navigating minefields when conducting entry operations in support of military objectives. Between 1990 and 1991, the Iraqi armed forces laid over 1,150 sea mines, precluding U.S. Forces from landing on Kuwait's coast. These minefields stretched over 80 miles, inflicting $27.5 million in damage to the USS Tripoli LPH-10 and the USS Princeton CG-59 (CNO 1991, Cornish 2003). In 2014, then Chairman of the Joint Chiefs of Staff General Martin Dempsey updated joint doctrine on port opening and entry operations in the littoral region, i.e., coastal shallows, to require that ships be prepared for the presence of armed opposition that is "... characterized by increasingly advanced area denial systems..." (Dempsey 2014). While most warfighting vessels cannot assume the mine clearing role of a larger MCMV, modern unmanned underwater vehicles (UUVs) provide the capability to locate and classify mines (Lin et al. 2008). These UUVs do not yet provide the robust clearance capabilities of an MCMV (Muljowidodo et al. 2009), but are highly effective at mapping minefields (Osborn 2017), and are specifically designed towards small-ship storage and launch (Kongsberg Maritime AS 2011). In the absence of advanced and/or economical clearance measures, the data collected by UUV Search-Classify-Map operations (Fletcher and Wernli 2003) must be exploited to provide Naval decision-makers with a risk analysis of navigating a given minefield.

In order to assist commanders during the Course of Action Analysis phase of the Joint Operations Planning Process (U.S. Joint Chiefs of Staff 2006), models and methods in this paper evaluate risk from individual mines within a minefield and, under the assumption that mines cannot be removed, constructs a method to find a path through the mapped minefield that minimizes the risk a ship incurs while traveling along such a path. We refer to this as the "minimum-risk routing model"; we assume that there are no ship navigational errors, that no modifications are made to the minefield once it is mapped (i.e., mines

remain stationary) and that ship movement is limited to a predefined Area of Operations (see Section 2.1) surrounding the field. We focus on minefields composed of "influence" mines, i.e., mines whose probability of detonation is proportional to a ship's proximity. Countries such as Italy and the former Soviet Union sell these mines at relatively low cost, making them a popular choice among navies with limited technological means (Freedberg 2012). We assume the location and threat radius (i.e., the farthest distance at which a ship risks damage from detonation) of each mine is determined using information gained from UUVs.

3.2.1 Previous Work

Minimum-risk routing methods generally fall into one of two categories: continuous or discrete. Continuous models typically rely on calculus-of-variation techniques and can involve a complex system of differential equations to plot an optimal curvilinear path (Karelahti et al. 2008, Mercer and Sidhu 2007, Zabarankin et al. 2002). Analytical approaches can be computationally prohibitive, and numerical formulations are solved as nonlinear programs with quadratic terms (Inanc et al. 2008).

Discrete models embed a static, directed graph $G(V, E)$ in a representation of the minefield in which vertices V represent waypoints and edges E denote possible segments for ship transit. The fact that paths in G can be made arbitrarily close to those of continuous models (Kim and Hespanha 2003, Muhandiramge et al. 2009), and are easily found through linear formulations makes discrete models well-suited for practical implementation. We recognize and address two of the major challenges faced by previous discrete approaches: over-accumulation of risk and excessive computation time.

The discrete approaches taken by Bekker and Schmid (2006), Boerman (1994) and Li (2009) all embed a directed graph in a representation of the minefield area and find a path that minimizes the sum of the risk along its edges. This sort of edge-additive model is attractive; an efficient shortest-path algorithm can provide swift solutions (Zwick 2001). An inherent issue with these models, however, is their vulnerability to over-accumulating

the risk of the path when considered in its entirety. Section 2.3 describes what Li (2009) terms "double-counting," which can lead to sub-optimal paths. Li limits, but does not eliminate, the chance that double-counting occurs. A more accurate and complete risk model is necessary for military commanders to make informed decisions.

3.2.2 Our Contribution

To address the shortfalls associated with traditional edge-additive models, we introduce a threat-additive model. By considering the risk from each mine rather than from each edge, our model provides mission commanders with superior information about their potential route. We use this information to generate G_R, a network representation of possible paths, and develop both a mixed integer program and an A* search algorithm to find a minimum-risk path through a given minefield. Although we employ exponential-time solution methods, we reduce problem sizes and runtimes by constructing an initial path that provides bounds on the minimum risk. In the case of our A* algorithm, we expand this heuristic to generate a unique graph G_V, tailored to the specific minefield, that may provide an improved solution. These initial solutions limit which paths must be evaluated and help remove many of the edges and vertices from G_R that cannot be part of a minimum-risk path.

3.3 Preliminaries

This section formally defines important military and mathematical concepts and introduces basic notation.

3.3.1 Network Representation of Area of Operations

We define the Area of Operations (AO) to be a square box in which all mines $m \in M$ are contained. We use a length (x-direction) and width (y-direction) of $\bar{d}$ that approximates the dimensions of entries to key ports essential to the domestic U.S. economy and necessary for timely logistical support of U.S. military actions abroad (Naval Studies Board 2001). We assume that the (x, y) coordinates and threat radius of each mine $m \in M$ have been determined via UUV Search-Classify-Map operations (Fletcher and Wernli 2003), and that

a ship must travel along the length of the AO under the assumption that physical boundaries exist at $y = 0$ and $y = \bar{d}$.

We define the *AO Network* as a weighted, directed graph $G = (V, E)$, with vertices $v \in V$ representing waypoints and directed edges $e \in E$ denoting possible segments to be traversed. Source and sink vertices s and t are placed outside the AO to allow ingress at any vertex $(0, y)$ and egress at $(\bar{d}, y)$, respectively. A possible path, E_{st}, is a connected set of edges that starts at s, terminates at t, and has no loops. While increases to the fidelity of the AO Network may provide additional, lower-risk paths, each such increase requires that $|V|$ and $|E|$ increase as well. We consider the topologies proposed by Bekker and Schmid (2006) and Li (2009) in Figure 3.1.

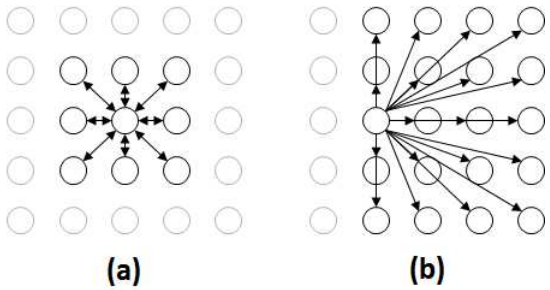


Figure 3.1: (a) Network topology used by Bekker and Schmid (2006); each vertex is evenly spaced in a rectilinear grid and connected to all immediate neighbors with bi-directional edges. (b) Network topology used by Li (2009); the vertex spacing is similar to (a), but removes all retrograde (i.e., right-to-left) edges and includes alternative, longer edges.

Both topologies in Figure 3.1 arrange V into a rectilinear grid and attempt to increase the tractability of the minimum-risk routing model without eliminating plausible, lower-risk paths. In Figure 3.1(a), Bekker and Schmid include bi-directional edges to account for paths that partially circumnavigate mines, but acknowledge that they must increase vertex spacing to control $|V|$ and $|E|$ in order to find "acceptable" solutions in a timely manner (Bekker and Schmid 2006).

Rather than focusing on $|V|$ or $|E|$, Li, in the topology of Figure 3.1(b), eliminates retrograde (i.e., right-to-left) edges and introduces alternative, longer edges. The omission of bi-directional edges may exclude lower-risk paths, but also eliminates difficult, acute turns. The longer edges reduce $|E_{st}|$ on average, and, thus, the time spent finding feasible solutions. Lastly, the topology in Figure 3.1(b) is a Directed Acyclic Graph (DAG). This guarantees a simple path and allows linear-time search algorithms that cannot be applied to the network in Figure 3.1(a) (Cormen 2009).

To incorporate the advantages of both topologies, we develop two separate AO Networks, G_R and G_V. The former, depicted in Figure 3.3(a), uses the simple rectilinear grid arrangement of V in Figure 3.1. Like Bekker and Schmid, we increase vertex spacing by using the realistic minimum turning radius of an 800-ton U.S. Navy ship (Li 2009). We then employ the DAG topology from Figure 3.1(b), but reduce $|E|$ by eliminating all vertical and "long" edges. Removing upward and downward edges restricts turns to a maximum of 45°, better modeling the turning circle of a ship (American Bureau of Shipping 2006) and more closely adhering to the straight-line assumptions of actuation curves (see Section 2.2). In lieu of *adding* long edges to E, we decrease the average $|E_{st}|$ using network reductions, introduced in Section 3.3, that *join* edges when possible and eliminate unnecessary vertices. Finally, source and sink vertices s and t are added to V, and E is updated to ensure that s connects to all potential entry vertices $(0, y)$, and that all potential exit vertices $(\bar{d}, y)$ connect to t.

The second network G_V, shown in Figure 3.3(b), uses a Voronoi diagram. In two dimensions, as illustrated in Figure 4.3, these diagrams take a set of "generators," each defined by an (x, y) coordinate, and partitions the plane into cells such that all points within a given cell are closer to the generator it contains than they are to any other generator (Edelsbrunner 2014).

We can partition a given minefield with a Voronoi diagram using the mine locations as generators. Representing each cell edge bi-directionally, we define $G_V(V, E)$ such that E

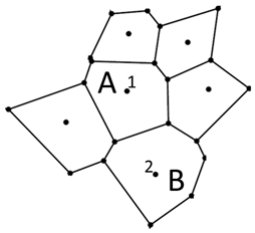


Figure 3.2: Voronoi diagram cell construction. All points in cell A are closer to generator 1 than they are to any other generator; the same is true for the points in cell B and generator 2. Every point along the edge separating cells A and B is equidistant to generators 1 and 2.

is the collection of these edges and V is the set of vertices at which they intersect. Both V and E must be slightly amended to account for locations where the Voronoi diagram intersects the bounding box of the AO. A vertex is added to V for each such intersection, and horizontal edges, representing movement along the top and bottom boundaries of the AO, are added to E between each pair of adjacent vertices along the top and along the bottom of the AO. Source and sink vertices s and t are added and connected to G_V in a manner similar to the way in which they are connected in G_R.

Depicting the AO Network via G_V has several advantages. Although G_V must be generated for each set of mines M, the complexity of this generation is relatively low at $\mathcal{O}(|M|\log|M|)$ (Fortune 1987). The construction of G_V also limits $|V|$ and $|E|$ to $\mathcal{O}(|M|)$ (Nandy 2009) while still allowing for circumnavigation. Among possible partitions, the construction also maximizes the minimum angle between edges (Edelsbrunner et al. 1992), reducing abrupt turns. Lastly, the minimum Closest Point of Approach (CPA) an edge makes to any mine is maximized, helping to reduce the risk (see Section 2.2) of a ship taking damage when traversing such an edge.

3.3.2 Probability of Actuation and Damage

We consider influence mines that detonate with a probability that is a function of the CPA of a ship. We define the threat radius R_m to be the maximum distance between a

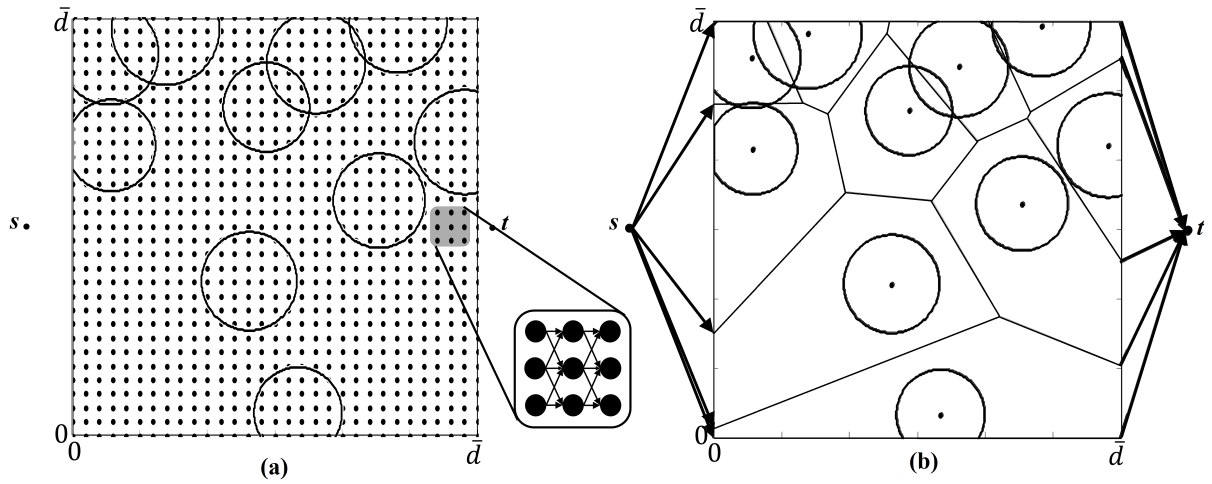


Figure 3.3: (a) G_R for a specific minefield. Circles represent threat radii and edges are omitted for clarity. Source and sink vertices s and t are connected to the AO by 30 left-to-right edges each. The inset shows the edge pattern followed throughout the interior of the AO. (b) G_V for the same minefield. Mines, centered within their circular threat radii, serve as the Voronoi generators. With the exception of those connected to s and t, all edges are bi-directional and their construction maximizes the minimum distance to neighboring mines.

ship and a mine at which unacceptable damage occurs if mine m detonates. Calculation of R_m is based on the type of ship navigating the AO along with data gathered by UUVs (Technical Manual 1996).

Defining CPA_{em} as the minimum distance between edge e and mine m, we can further specify p_{em} as the probability of mine m detonating and a ship incurring unacceptable damage if it travels along edge e. For values of CPA_{em} greater than R_m, detonation is technically possible, but at such a distance that the ship is assumed to incur no damage; therefore, $p_{em} \equiv 0$. When CPA_{em} is within a mine's threat radius, p_{em} is determined via an actuation curve derived from operational test data (U.S. Navy 1959). For simplicity, we choose the triangular actuation curve in Figure 3.4 defined by

$$p_{em} \equiv \begin{cases} \dfrac{R_m - |CPA_{em}|}{R_m} & \text{if } |CPA_{em}| < R_m, \\ 0 & \text{if } |CPA_{em}| \geq R_m. \end{cases} \tag{3.1}$$

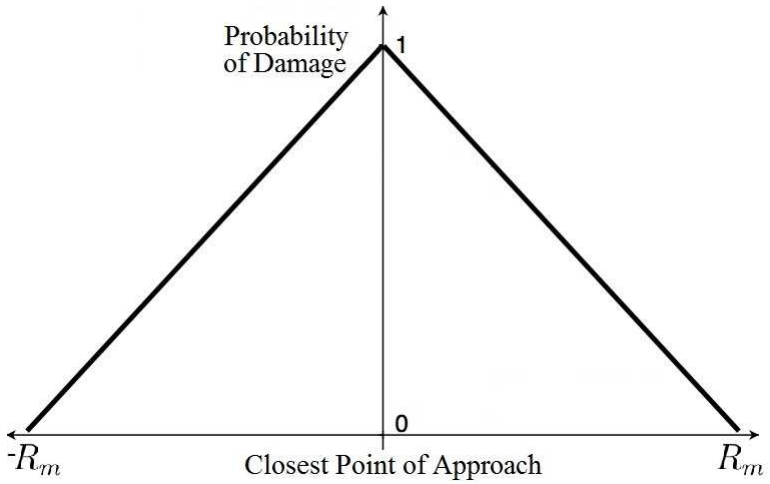


Figure 3.4: Triangle actuation curve (Eagle 2008). The probability p_{em} of a ship incurring unacceptable damage from the detonation of mine m while traversing edge e is depicted along the y-axis as a function of the closest point of approach, CPA_{em}, made by edge e, depicted along the x-axis. Negative and positive CPA_{em} values indicate the passing of a mine on the port and starboard sides, respectively.

Detonation does not necessarily occur at the CPA; thus, an actuation curve estimates the probability that detonation occurs at *some point* during the straight-line approach of a ship (U.S. Navy 1959). Identical CPA values, therefore, yield the same probability of detonation. Our solutions, however, may violate the straight-line assumption since a minimum-risk path, comprised of line segments, may deviate from an infinitely long, straight line. The construction of G_R and G_V attempts to mitigate this by reducing the number and sharpness of turns within E_{st}. Additionally, we introduce a distance penalty in Section 3 that avoids unnecessary turns and encourages straight paths.

3.3.3 Risk Accumulation

For a ship traversing a path E_{st}, we consider its mission to be a success if it survives, i.e., avoids unacceptable damage from *all* mines within the AO. Thus, if $\mathcal{S}_{E_{st}}$ denotes the probability of survival for a ship traversing path E_{st}, the risk of the path is defined as:

$$\mathcal{R}_{E_{st}} \equiv 1 - \mathcal{S}_{E_{st}}. \tag{3.2}$$

Post-war analysis (U.S. Congress 1992) and U.S. wargaming (Ocean Studies Board 2000) concentrate on widely available influence mines such as the Soviet KMD and Italian MANTA. Although these mines can be programmed for acoustic or magnetic sensitivity, they do not share information (RWM Italia S.p.A. 2012). Each detonation, therefore, is an independent event. Subsequently, if we define M_e as the set of mines which impinge upon edge e, the probability of survival along e can be calculated as the product of non-damage probabilities incurred from each individual mine:

$$\mathcal{S}_e = \prod_{m \in M_e} (1 - p_{em}). \tag{3.3}$$

The probability $\mathcal{S}_{E_{st}}$ of a ship surviving traversal of path E_{st} can be calculated via Equation (3.3) by considering E_{st} as a single edge and determining the probability of damage using the CPA of each mine to E_{st}. In a discretized model, however, our path is represented as a *set* of edges and must aggregate the probability of survival along multiple edges.

To do so, a logarithmic transformation (Shorack 1964) linearizes the product in (3.3):

$$\mathcal{S}_e = \exp\left(\sum_{m \in M_e} \ln(1 - p_{em})\right). \tag{3.4}$$

In the traditional "edge-additive" model, each edge traversal is treated as an independent event. The probability of survival along multiple edges is simply compounded in the

same way (3.3) compounds the probability of surviving passage beyond multiple mines. As such, it can be similarly calculated and linearized as before to yield $\widehat{\mathcal{S}}_{E_{st}}$, the *edge-additive* probability of survival along a path E_{st}:

$$\widehat{\mathcal{S}}_{E_{st}} \equiv \exp\left(\sum_{e\in E_{st}}\sum_{m\in M_e} \ln(1-p_{em})\right). \tag{3.5}$$

Because, for our application, $p_{em} \in [0,1)$, the expression $\ln(1-p_{em})$ is non-positive for all p_{em}. Thus, we define non-negative parameters r_{em} and r_e such that

$$r_{em} \equiv -\ln(1-p_{em}), \tag{3.6}$$

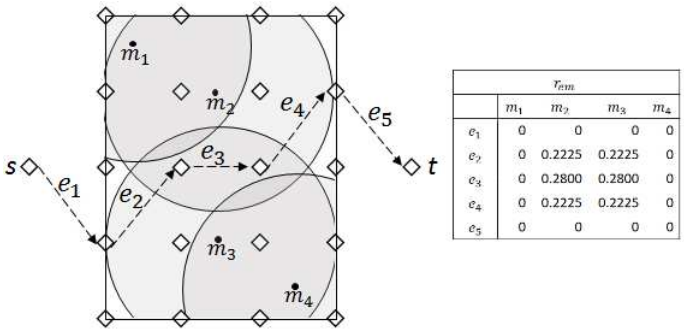
$$r_e \equiv \sum_{m\in M_e} r_{em}. \tag{3.7}$$

Substituting r_e into (3.5), we replace $\mathcal{S}_{E_{st}}$, in Equation (3.2), with $\widehat{\mathcal{S}}_{E_{st}}$ and define the edge-additive risk for a path E_{st}:

$$\widehat{\mathcal{R}}_{E_{st}} \equiv 1 - \exp\left(-\sum_{e\in E_{st}} r_e\right). \tag{3.8}$$

By assigning r_e as a non-negative "weight" to each edge $e \in E$, a shortest-path algorithm can find a path E_{st} that minimizes $\sum_{e\in E_{st}} r_e$ and, thus, $\widehat{\mathcal{R}}_{E_{st}}$. While this edge-additive approach provides fast solutions, $\widehat{\mathcal{R}}_{E_{st}}$ may over-estimate $\mathcal{R}_{E_{st}}$ and result in a sub-optimal path. Figure 3.5 demonstrates how summing r_e across all edges on a path can overestimate $\mathcal{R}_{E_{st}}$. Ignoring edges e_1 and e_5 which incur no risk, we can calculate $\widehat{\mathcal{R}}_{E_{st}}$ for the path E_{st} in Figure 3.5:

$$
\begin{aligned}
\widehat{\mathcal{R}}_{E_{st}} &= 1 - \exp\left(-\left(r_{e_2} + r_{e_3} + r_{e_4}\right)\right) \\
&= 1 - \exp\left(-\left(\left(r_{e_2,m_2} + r_{e_2,m_3}\right) + \left(r_{e_3,m_2} + r_{e_3,m_3}\right) + \left(r_{e_4,m_2} + r_{e_4,m_3}\right)\right)\right) \\
&= 1 - \exp\left(-\left(0.2225 + 0.2225 + 0.28 + 0.28 + 0.2225 + 0.2225\right)\right) \qquad (3.9) \\
&= 1 - \exp(-1.47) \\
&\approx 0.77.
\end{aligned}
$$



r_{em}	m_1	m_2	m_3	m_4
e_1	0	0	0	0
e_2	0	0.2225	0.2225	0
e_3	0	0.2800	0.2800	0
e_4	0	0.2225	0.2225	0
e_5	0	0	0	0

Figure 3.5: Risk accumulation along a path. In this small model of an AO that employs topology from G_R, only mines m_2 and m_3 impinge upon the path indicated by the dashed edges. The risk of traversing the path is determined by how closely it approaches both mines. This point occurs along edge e_3 and, as calculated by Equation (3.10), incurs a total $\mathcal{R}_{E_{st}}$ of 0.43. The edge-additive model, however, "double-counts" risk on edges e_2 and e_4 as the path approaches and recedes from the *CPA*, and Equation (3.9) overestimates $\mathcal{R}_{E_{st}}$ at 0.77.

To accurately calculate $\mathcal{R}_{E_{st}}$, we must consider the path in its entirety. As such, we use the CPA of the *path* to mines m_2 and m_3. Edge e_3 represents the CPA of both mines to the path, and leads to a much lower risk:

$$
\begin{aligned}
\mathcal{R}_{E_{st}} &= 1 - \exp(-r_{e_3}) \\
&= 1 - \exp(-(r_{e_3,m_2} + r_{e_3,m_3})) \\
&= 1 - \exp(-(0.28 + 0.28)) \\
&= 1 - \exp(-0.56) \\
&\approx 0.43.
\end{aligned} \tag{3.10}
$$

Li (2009) recognizes the overestimation, exemplified by Equation (3.9), and that this double-counting of risk from mines can lead to a sub-optimal solution. Li attempts to mitigate double-counting by introducing the topology with longer edges illustrated in Figure 3.1(b). By increasing the chance that edges in E_{st} span the entire threat radius of a mine, the chance of double-counting that mine's risk is reduced. As long as multiple edges can intersect a single threat radius, however, the possibility of double-counting cannot be eliminated completely.

In Equation (3.10), accurate calculation of $\mathcal{R}_{E_{st}}$ depends upon considering individual mines rather than edges. Instead of the edge-additive weight defined in Equation (3.7), we let E_m be the set of edges upon which a mine m impinges and use r_{em}, defined in Equation (4.1), to construct a *threat*-additive weight w_m such that

$$
w_m \equiv \max_{e \in E_m \cap E_{st}} r_{em}. \tag{3.11}
$$

Because w_m is uniquely defined for each mine, summing it across all $m \in M$ accurately calculates $\mathcal{R}_{E_{st}}$ without double-counting:

$$
\mathcal{R}_{E_{st}} = 1 - \exp\left(- \sum_{m \in M} w_m \right). \tag{3.12}
$$

Although Equation (3.12) eliminates instances of double-counting and ensures accurate risk calculation, the construct in Equation (3.11) precludes the use of shortest-path algo-

rithms to minimize $\mathcal{R}_{E_{st}}$. The following section develops the formulations and methods necessary to identify a path that minimizes (3.12).

3.4 Formulations and Methods

This section explicitly defines integer programs (IPs) and algorithms for finding paths that minimize the risk of traversal, as defined in Equations (3.8) and (3.12). First, we formulate the traditional edge-additive model as an IP. Next, we present two separate, threat-additive IP formulations to identify the minimum-risk path through a given AO Network. Additionally, we develop a novel A* algorithm (Hart et al. 1968). To reduce computational time, we introduce several heuristics that produce initial solutions, reduce the size of the network, and bound the minimum risk.

3.4.1 Edge-Additive Formulation (SP)

Using notation from the previous section, we formulate the traditional edge-additive model as a linear program. To minimize $\widehat{\mathcal{R}}_{E_{st}}$, as defined in (3.8), it suffices to select an s-t path to minimize $\sum_{e \in E_{st}} r_e$. A binary decision variable x_e represents whether a particular edge e is included in the path E_{st}. To avoid unnecessary turns that violate the straight-line assumptions made by the actuation curve defined in Section 2.2, a small distance penalty ϵd_e is included in the objective function (3.13), where $0 < \epsilon \ll 1$ and where d_e represents the length of edge e. (We use $\epsilon = 10^{-7}$ in our computation.) For the solutions we find, the penalty has a negligible effect on total risk. Constraint (3.14) is a standard flow balance requirement that ensures E_{st} is simple and connected; sets $\overleftarrow{E}_v$ and $\overrightarrow{E}_v$ specify the edges directed into and out of v, respectively. Total unimodularity of Constraint (3.14) (Tutte 1958) also ensures the integrality of x_e and, thus, it suffices that Constraint (3.15) require only that $0 \leq x_e \leq 1$.

$$\textbf{SP}: \min_{x} \quad \sum_{e \in E} \Big(r_e x_e + \epsilon d_e x_e \Big) \tag{3.13}$$

$$\text{s.t.} \quad \sum_{e \in \overleftarrow{E}_v} x_e - \sum_{e \in \overrightarrow{E}_v} x_e = \begin{cases} 1 & \text{if } v = s \\ -1 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall v \in V \tag{3.14}$$

$$x_e \in \big[0, 1\big] \qquad \forall e \in E \tag{3.15}$$

3.4.2 Threat-Additive Formulations

We introduce two IP formulations to solve the minimum-risk routing model using the threat-additive approach developed in Section 2.3. The first formulation, **TMR-1**, minimizes the maximum risk a mine imposes across the edges in a path. The second, **TMR-2**, admits the same set of integer-feasible solutions as the first, but possesses a stronger LP relaxation than **TMR-1**.

3.4.2.1 Threat-Additive Minimum-Risk Formulation 1

To find a path that minimizes $\mathcal{R}_{E_{st}}$, it suffices to minimize the summation $\sum_{m \in M} w_m$ found in Equation (3.12). Parameters ϵ, d_e, variable x_e, and Constraint (3.14) are all defined as in **SP**. Constraint (3.17) is a standard min-max construct to ensure that w_m is calculated according to its definition in Equation (3.11). Constraint (3.18) restricts x_e to binary values. Because r_{em} and x_e are non-negative for all $e \in E, m \in M$, Constraint (3.17) also restricts w_m to non-negative values.

$$\textbf{TMR-1}: \min_{w,x} \quad \sum_{m \in M} w_m + \epsilon \sum_{e \in E} d_e x_e \tag{3.16}$$

$$\text{s.t.} \quad (3.14)$$

$$w_m \geq r_{em} x_e \qquad \forall e \in E, m \in M_e \tag{3.17}$$

$$x_e \ \in \{0, 1\} \qquad \forall e \in E \tag{3.18}$$

3.4.2.2 Threat-Additive Minimum-Risk Formulation 2

Using a technique introduced by Nehme (2009), we can strengthen the LP relaxation in **TMR-1** while formulating an equivalent integer program. Starting with a particular type of step inequalities (Morton et al. 2007), Nehme presents a reformulation based on an extended variable formulation that exploits the so-called mixing set structure that appears in **TMR-1** (see Luedtke et al. (2010) and Miller and Wolsey (2003)). The extended variable formulation results in a larger model, with additional decision variables and constraints, but has a stronger linear programming relaxation, which can decrease the required computational effort.

For each $m \in M$, we define a bijection $\mathcal{B}_m : E_m \to J_m$, such that $j \in J_m = \{1, 2, \ldots, |E_m| + 1\}$ represents the edge $e \in E_m$ with the j^{th} largest r_{em} value, and $j = |E_m| + 1$ represents a dummy edge $\hat{e}$ such that $r_{\hat{e}m} \equiv 0$. Under this construction, $\mathcal{B}_m$ satisfies

$$r_{em} \leq r_{e'm} \quad \forall m \in M, e' \in E_m, e \in E_m \cup \{\hat{e}\} : \mathcal{B}_m(e') \leq \mathcal{B}_m(e). \tag{3.19}$$

We also introduce a binary decision variable y_{jm}, which assumes a value of one if a ship traverses any edge that incurs more risk from mine m than the edge represented by $j \in J_m$, and 0 otherwise:

$$x_e \leq y_{jm} \quad \forall m \in M, e \in E_m, j \in J_m : \mathcal{B}_m(e) < j. \tag{3.20}$$

Because Equation (3.20) generates $\mathcal{O}\left(\sum_{m \in M} |E_m|^2\right)$ additional constraints, we reformulate Equation (3.20) as two separate constraints:

$$x_e \leq y_{jm} \quad \forall m \in M, e \in E_m, j \in J_m : \mathcal{B}_m(e) = j - 1, \tag{3.21}$$

$$y_{jm} \leq y_{j+1,m} \quad \forall m \in M, e \in E_m, j = 2, 3, \ldots, |E_m|, \tag{3.22}$$

which collectively generate only $\mathcal{O}\left(\sum_{m \in M} |E_m|\right)$ constraints, by contrast. For convenience, we define the parameter Δ_{jm} such that

$$\Delta_{jm} = r_{e'm} - r_{em} \; \forall m \in M, e \in E_m : j = \mathcal{B}_m(e) = \mathcal{B}_m(e') + 1. \tag{3.23}$$

The ordering imposed by (3.19), along with Constraints (3.21) and (3.22), allow w_m, defined in Equation (3.11), to be reformulated as:

$$w_m = \sum_{j=2}^{|E_m|} \Delta_{jm} y_{jm} \; \forall m \in M. \tag{3.24}$$

To understand how Equation (3.24) appropriately represents w_m, consider a mine m that impinges on five edges: $e_1, \ldots, e_5$. For a particular path that only includes edges e_3 and e_5 that incur the third and fifth highest risk from m, respectively, Constraint (3.21) forces y_{4m} and y_{6m} to 1, Constraint (3.22) sets $y_{5m} = 1$, and Equation (3.24) becomes

$$\begin{aligned} w_m = {} & (r_{e_1,m} - r_{e_2,m}) \cdot 0 + (r_{e_2,m} - r_{e_3,m}) \cdot 0 \\ & + (r_{e_3,m} - r_{e_4,m}) \cdot 1 + (r_{e_4,m} - r_{e_5,m}) \cdot 1 + (r_{e_5,m} - 0) \cdot 1, \end{aligned}$$

which reduces to $r_{e_3,m}$, the maximum risk incurred by the path. Equation (3.24) accounts for the max operator in Equation (3.11), and allows reformulation of **TMR-1** without w_m as:

$$\textbf{TMR-2}: \min_{x,y} \quad \sum_{m \in M} \sum_{j=2}^{|E_m|} \Delta_{jm} y_{jm} + \epsilon \sum_{e \in E} d_e x_e \tag{3.25}$$

$$\text{s.t.} \quad (3.14), (3.18), (3.21), (3.22)$$

$$y_{jm} \in [0,1] \qquad \forall m \in M, j \in J_m. \tag{3.26}$$

Constraint (3.14) maintains path continuity. Constraints (3.21) and (3.22) force the objective function to eliminate double-counting and correctly accumulate risk. The combination of Constraints (3.21) and (3.22) with the binary restriction imposed by Constraint (3.18) and the sense of the objective function, i.e., minimization, allows the variable in Constraint (3.26) to remain continuous. Although it is possible to relax Constraint (3.18) by enforcing integrality upon the y_{jm} variable in Constraint (3.26), this tends to be impractical. Unlike the number of x_e variables, which remains equal to $|E|$, the number of y_{jm} variables is a function of $|M|$ and the average threat radius R_m. Using the instance parameters defined in Section 4, the number of y_{jm} variables is approximately $(0.045)|M| \cdot |E|$, and quickly surpasses $|E|$ for $|M| \geq 30$.

3.4.3 Minimum-Risk Path Algorithm

While a commercial solver such as CPLEX can provide relatively fast and accurate solutions to **TMR-1** and **TMR-2**, situations occur during military planning in which a non-commercial option is advantageous. To this end, we construct an algorithm based on Byers and Waterman (1984) in C++ using the Library for Efficient Modeling and Optimization in Networks (LEMON) from the open source Computational Infrastructure for Operations Research (COIN-OR) (Dezső et al. 2011). To solve the threat-additive model, this Minimum-Risk Path Algorithm (***MRPA***) limits the paths that must be evaluated by setting bounds on the risk associated with an optimal path using heuristics ***H*** (Section 3.3.1) and ***VGB*** (Section 3.3.3), and by reducing the size of the AO Network G with preprocessing steps ***P*** and ***R*** (Section 3.3.4). In defining pseudocode for these functions, we assume all parameters are passed by reference, and that a network G includes V, E, and

all associated parameters and subsets. Similarly, M includes mine locations, their threat radii, and, together with G, implies knowledge of any derived parameter such as r_{em}.

3.4.3.1 Risk-Bounding Heuristic (*H*)

The heuristic ***H*** uses a modified version of **SP**, implemented in LEMON, to find an initial path $\widehat{E}_{st}$. In lieu of r_e, ***H*** uses $\underline{r}_e$, defined as

$$\underline{r}_e \equiv \sum_{m \in M_e} \frac{r_{em}}{n_m}, \tag{3.27}$$

where $n_m \leq |E_m|$ is the maximum number of edges upon which mine m can impinge in any s-t path. Weighting r_{em} by the reciprocal of n_m does not eliminate the double-counting associated with a shortest-path algorithm, but ensures that for any path E_{st},

$$\sum_{e \in E_m \cap E_{st}} \underline{r}_e \leq w_m \tag{3.28}$$

for each $m \in M$. Like **TMR-1**, in order to minimize $\mathcal{R}_{E_{st}}$ it suffices that ***MRPA*** minimizes $\sum_{m \in M} w_m$. For convenience, we introduce notation $\underline{r}_{E_{st}}$ and $r_{E_{st}}$ such that:

$$\underline{r}_{E_{st}} \equiv \sum_{e \in E_{st}} \underline{r}_e, \tag{3.29}$$

$$r_{E_{st}} \equiv \sum_{m \in M} w_m, \tag{3.30}$$

where w_m is defined in Equation (3.11) with respect to E_{st}. Because the path $\widehat{E}_{st}$, as determined by ***H***, minimizes $\underline{r}_{E_{st}}$, it follows from (3.28) that $\underline{r}_{\widehat{E}_{st}}$ defines a lower bound on the minimum $r_{E_{st}}$. Likewise, ***H*** defines an upper bound $r_{\widehat{E}_{st}}$ as measured by Equation (3.30). In addition to providing bounds for ***MRPA***, the linear program **SP** can efficiently generate $\widehat{E}_{st}$, using $\underline{r}_e$ in lieu of r_e and, thus, provide **TMR-1** and **TMR-2** with an initial solution. Lastly, in addition to $\underline{r}_{E_{st}}$, ***H*** finds the "shortest" path from v to t for each $v \in V$

and computes a similar lower bound $\underline{r}_{E_{vt}}$. For any s-t path which passes through v, at least $\underline{r}_{E_{vt}}$ of the sum defined by $r_{E_{st}}$ must be incurred traveling from v to t. Pseudocode for $\boldsymbol{H}$ is provided in Figure 3.6.

Function $\boldsymbol{H}(G, M)\{$
`// Apply weights to edges and find initial solution`
foreach $e \in E$ **do** $\underline{r}_e \leftarrow \sum_{m \in M_e} \frac{r_{em}}{n_m}$;
$\widehat{E}_{st} \leftarrow$ "shortest" s-t path (via Dijkstra with $\underline{r}_e$);
`// Define lower and upper bounds on risk`
$\underline{r}_{\widehat{E}_{st}} \leftarrow \sum_{e \in \widehat{E}_{st}} \underline{r}_e$;
$r_{\widehat{E}_{st}} \leftarrow \sum_{m \in M} w_m$ for the path $\widehat{E}_{st}$;
`// Define lower bounds for partial paths`
foreach $v \in V$ **do**
  $E_{vt} \leftarrow$ "shortest" v-t path (via Dijkstra via $\underline{r}_e$);
  $\underline{r}_{E_{vt}} \leftarrow \sum_{e \in E_{vt}} \underline{r}_e$
end
}

Figure 3.6: Pseudocode for $\boldsymbol{H}$. The heuristic defines an initial path and sets bounds on the risk incurred by a minimum-risk path. Lower bounds for each $v - t$ path are stored at vertex v

3.4.3.2 Implicit Path Enumeration

Unlike the IP formulations, ***MRPA*** finds a route that minimizes risk by calling a recursive function (Figure 3.7) that builds a path from s to t one edge at a time. At each vertex u along a sub-path E_{su}, ***MRPA*** computes $r_{E_{su}}$. To decide whether the subpath should be extended to vertex v along edge $e = (u, v)$, ***MRPA*** uses a two-step decision process. First, ***MRPA*** checks the following condition:

$$r_{E_{su}} + \underline{r}_e + \underline{r}_{E_{vt}} < r_{\widehat{E}_{st}}. \tag{3.31}$$

If (3.31) is false, extending E_{su} along e cannot possibly result in a path with lower risk than the incumbent path $\widehat{E}_{st}$, and ***MRPA*** continues evaluating the condition for edges

in $\overrightarrow{E}_u$, i.e., the edges leaving u. If (3.31) holds and $v \notin \widehat{E}_{st}$, ***MRPA*** calculates $r_{E_{sv}}$ and checks the second condition:

$$r_{E_{sv}} < r_{\widehat{E}_{st}}. \tag{3.32}$$

If (3.32) is true, ***MRPA*** extends the sub-path along edge e. As previously, if (3.32) is false, we have determined that extending our sub-path along e cannot result in an s-t path with less risk than the incumbent, and ***MRPA*** proceeds to evaluate the next edge in $\overrightarrow{E}_u$.

```
Function ExtendPath(G, M, u, E_su){
// Evaluate edges leaving u, exclude vertices already on sub-path
foreach e = (u, v) ∈ E→_u such that E→_v ∩ E_su = ∅ do
    // Check necessary conditions for extending sub-path
    E_sv ← E_su ∪ {e};
    if (r_{E_su} + r_e + r_{E_vt} < r_{Ê_st}) and (r_{E_sv} < r_{Ê_st}) then
        if v = t then
            // Update incumbent path and the upper bound on risk
            Ê_st ← E_sv;
            r_{Ê_st} ← r_{E_sv};
        else
            // Attempt to complete s-t path
            ExtendPath(G, M, v, E_sv);
        end
    end
end
}
```

Figure 3.7: Pseudocode for improving the current sub-path E_{su}. The call to **ExtendPath** recursively stores all interim solutions until all feasible improvements have been exhausted and $\widehat{E}_{st}$ represents a minimum-risk path from s to t through G.

MRPA continues enumerating and evaluating sub-paths until E_{su} meets one of two conditions. First, if ***MRPA*** cannot successfully extend E_{su} along any of the edges in $\overrightarrow{E}_u$, it returns to u', the penultimate vertex along E_{su}, and resumes evaluation of the edges in

$\overrightarrow{E}_{u'}$ in search of a viable path extension. Otherwise, if ***MRPA*** successfully extends its sub-path to vertex t, then it has found a new s-t path with less risk than the incumbent. In this case, ***MRPA*** replaces the incumbent path $\widehat{E}_{st}$ with the current sub-path and updates $r_{\widehat{E}_{st}}$ to reflect the lowered risk. As with the first condition, ***MRPA*** then retreats to the previous vertex, evaluating alternative paths relative to the new, improved upper bound.

3.4.3.3 Voronoi Graph Bound (*VGB*)

While finding minimum-risk paths through G_V proves to be much faster than searching G_R, paths in G_V are not guaranteed to match the risk of those associated with G_R. The simple, two-mine example in Figure 3.8 demonstrates.

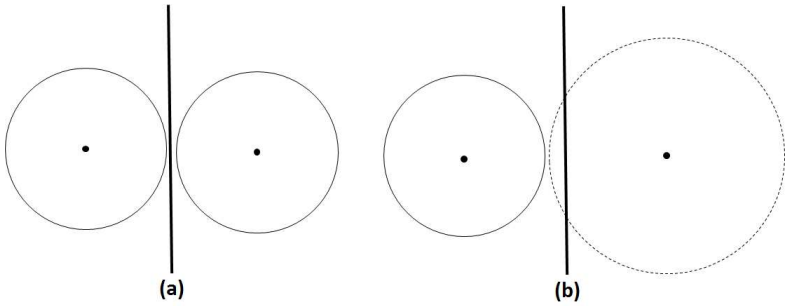


Figure 3.8: (a) With equal threat radii (solid circles) the Voronoi edge is equidistant to both mines and represents the lowest-risk path. (b) With a non-equal threat radii (dashed circle), the Voronoi edge now intersects the larger circle, even though a zero-risk route exists.

As defined in Section 2.1, the construction of G_V maximizes the minimum distance from its edges to each neighboring mine, which is equivalent to minimizing the maximum probability of damage if, as in Figure 3.8(a), the threat radius R_m of each mine is equal. In the instance of non-equal threat radii, however, this equivalence ceases to hold. As the number of mines and the variation in their radii increases, situations such as the one in Figure 3.8(b) may occur in which a ship traversing an edge in G_V enters the larger of two threat radii when a lower or zero-risk path exists.

Because we allow the threat radii in our test scenarios to vary between 300 and 400 yards (see Section 4), we may encounter situations such as the one depicted in Figure 3.8(b) as minefield density increases. Replacing G_V with a *weighted* Voronoi diagram may eliminate such discrepancies; but doing so without significantly sacrificing the favorable complexity G_V possesses (Har-Peled and Raichel 2014) requires further research and may result in complicated impractical paths. Thus, Figure 3.9 defines a *heuristic* ***VGB***, which finds an s-t path by applying ***H*** and **ExtendPath** to G_V. To ensure ***MRPA*** is able to match the solutions found by our IPs, the algorithm searches G_R for a minimum-risk path, but applies ***VGB*** to check if the initial path $\widehat{E}_{st}$, found by $\boldsymbol{H}(G_R, M)$, can be replaced with a lower-risk path from G_V to reduce the upper bound $r_{\widehat{E}_{st}}$ and the number of sub-paths that **ExtendPath** must explore in G_R.

Function $\boldsymbol{VGB}(M)\{$
Generate G_V using M locations;
`// Find initial path through` G_V
$\boldsymbol{H}(G_V, M)$;
`// Reduce the AO Network (See Section 3.3.4)`
$\boldsymbol{P}(G_V, M)$;
$\boldsymbol{R}(G_V, M)$;
`// Find minimum-risk path through` G_V
ExtendPath$(G_V, M, s, \emptyset)$;
$\}$

Figure 3.9: Pseudocode for ***VGB***. The path $\widehat{E}_{st}$ and its associated risk are set according to the minimum-risk path within G_V.

3.4.3.4 Network Reductions (*P* and *R*)

In addition to ***VGB***, we tailor two prepossessing procedures ***P*** (Carlyle et al. 2008) and ***R*** (Carlyle et al. 2009), to remove edges and vertices from the AO Network that either cannot be part of a minimum-risk solution or that are unnecessary. After ***H*** calculates

$\underline{r}_{E_{vt}}$ for each $v \in V$, $\boldsymbol{P}$ defines a similar lower bound $\underline{r}_{E_{sv}}$ on the risk from s to each vertex. Then, for each $v \in V$, $\boldsymbol{P}$ checks whether

$$\underline{r}_{E_{sv}} + \underline{r}_{E_{vt}} \geq r_{\widehat{E}_{st}}. \tag{3.33}$$

No $v \in V$ such that (3.33) holds can lie on a minimum-risk path and, thus, $\boldsymbol{P}$ removes v and each adjacent edge $e \in \overrightarrow{E}_v \cup \overleftarrow{E}_v$ from the AO Network. Additionally, $\boldsymbol{P}$ identifies and removes each individual edge $e \in E$ such that $r_e \geq r_{\widehat{E}_{st}}$. Lastly, $\boldsymbol{P}$ removes any "dead ends" (i.e., vertices v such that $\overrightarrow{E}_v = \emptyset$ or $\overleftarrow{E}_v = \emptyset$) it may have created.

In contrast to $\boldsymbol{P}$, which removes "high-risk" vertices and edges, $\boldsymbol{R}$ searches for "risk-free" vertices v such that

$$M_e = \emptyset \quad \forall e \in \overleftarrow{E}_v \cup \overrightarrow{E}_v. \tag{3.34}$$

For every $v \in V$ such that (3.34) is true, $\boldsymbol{R}$ removes v and replaces each inbound-outbound pair of edges with a single, zero-risk edge. Because $\boldsymbol{R}$ is rerun until G is reduced as much as possible, each new edge created in this way is only added to G if it is shorter than any such edge already in E.

Both $\boldsymbol{P}$ and $\boldsymbol{R}$ are applied to our IP formulations in the form of indexing restrictions, and to ***MRPA*** using code defined in the appendix. For ***MRPA***, every edge and vertex removed by $\boldsymbol{P}$ enables the elimination of an *exponential* number of paths ***MRPA*** must evaluate. Each time $\boldsymbol{R}$ successfully joins edges, ***MRPA*** is able to update $\widehat{E}_{st}$ with fewer calls to **ExtendPath** and, thus, more frequently improve the upper bound $r_{\widehat{E}_{st}}$. Using our heuristics, preprocessing and implicit search, Figure 3.10 formally defines the procedure ***MRPA*** follows to find a minimum-risk s-t route through a given minefield.

3.5 Computational Results and Analysis

We generate **SP**, **TMR-1**, and **TMR-2** using the AMPL modeling language (Fourer et al. 2003) and solve both via CPLEX 12.6.2 (IBM 2010) running on Intel Xeon 2.60GHz processors using 32 GB of RAM; a 1% optimality gap holds for $\mathcal{R}_{E_{st}}$. We code ***MRPA*** in

C++ LEMON running on an Intel i5 1.60GHz with 8 GB of RAM. We set the dimensions of our AO such that $\bar{d} = 3,000$ yards and run tests for $|M| \in \{10, 20, \ldots, 100\}$, as minefields of this size and density accurately reflect the range of historical real-world encounters (Finlan 2004). We generate 30 instances for each minefield density and, for each instance, randomly determine mine locations within the AO by choosing x- and y-coordinates from a uniform distribution on $[0, \bar{d}]$.

Function ***MRPA***$(G_R, M)\{$
// Find initial path through G_R
H(G_R, M);
$\widehat{E}^R_{st} \leftarrow \widehat{E}_{st}$;
$r_{\widehat{E}^R_{st}} \leftarrow r_{\widehat{E}_{st}}$;
// Generate and find the minimum-risk path through G_V
VGB(M);
$\widehat{E}^V_{st} \leftarrow \widehat{E}_{st}$;
$r_{\widehat{E}^V_{st}} \leftarrow r_{\widehat{E}_{st}}$;
// Choose best initial path and upper bound
if $r_{\widehat{E}^V_{st}} < r_{\widehat{E}^R_{st}}$ **then**
  $\widehat{E}_{st} \leftarrow \widehat{E}^V_{st}$;
  $r_{\widehat{E}_{st}} \leftarrow r_{\widehat{E}^V_{st}}$;
end
else
  $\widehat{E}_{st} \leftarrow \widehat{E}^R_{st}$;
  $r_{\widehat{E}_{st}} \leftarrow r_{\widehat{E}^R_{st}}$;
end
// Reduce the AO Network
P(G_R, M);
R(G_R, M);
// Find the minimum-risk path in G_R
ExtendPath$(G_R, M, s, \emptyset)$;
$\}$

Figure 3.10: Pseudocode for ***MRPA***. Upon completion, $\widehat{E}_{st}$ represents the minimum-risk route possible among all s-t paths in both G_R (found via **ExtendPath**) and G_V (found via ***VGB***).

Under carefully controlled scenarios, the range at which a ship incurs specific types of damage can be calculated with relative accuracy and is small compared to the distance at which a ship might reasonably cause a mine to detonate (Szturomski 2015). Because we make no assumptions about the speed, depth, or rigidity of a ship's hull, however, and measure risk on the premise that *any* damage is unacceptable, our testing assumes an average threat radius of 350 yards based on reasonable boundaries of *actuation*, found during operational testing (Maldonado 1996). Unlike previous models (Bekker and Schmid 2006, Li 2009), we allow for variation among our threat radii and choose each radius from a uniform distribution on $[300, 400]$.

For each $|M|$, we compare the paths found in G_R by the edge- and threat-additive IPs based on their average risk and solution times. For each test instance, G_R is static, as defined in Section 2.1, and initially possesses 963 vertices in a rectilinear grid with 100 yard spacing connected by 2,793 edges. Solutions found by ***MRPA*** are compared using the same measures, but may include additional lower-risk paths found in G_V, generated by ***VGB***, and, thus, not available to our IP formulations. The number of edges and vertices in G_V varies for each instance we test, but remains below 300 and 200, respectively, in all cases.

3.5.1 Solution Quality

Although the edge-additive model **SP** overestimates $\mathcal{R}_{E_{st}}$ for a path, the correct value for a path found by **SP** can be evaluated ex posteriori for direct comparison to those produced by our threat-additive model. As they provide equivalent solutions, we refer to **TMR-1** and **TMR-2** collectively as **TMR** for the purposes of this section.

Table 3.1 summarizes the average $\mathcal{R}_{E_{st}}$ for optimal paths, as found by **SP** and **TMR**, across all 30 instances of each $|M|$. All paths found by **TMR** possess $\mathcal{R}_{E_{st}}$ equal to or less than paths found by **SP**. Furthermore, the paths found by **SP** are sub-optimal relative to those found by **TMR** in 130 of the 300 test cases. In these instances, our threat-additive model finds paths that decrease the probability of incurring damage by an average of 0.07.

Table 3.1: Risk $\mathcal{R}_{E_{st}}$ associated with optimal paths found by edge-additive (**SP**) and threat-additive (**TMR**) models. The fixed dimensions of the AO imply that each value of $|M|$ determines a unique minefield density, with columns 2 and 3 representing the average of the minimum-risk across all 30 instances of that particular density. Column 4 indicates the percentage of instances in which **SP** finds paths that prove to be sub-optimal.

$\|M\|$	$\mathcal{R}_{E_{st}}$		Sub-Optimal Instances
	SP	**TMR**	
10	0	0	0%
20	0.07	0.05	10%
30	0.26	0.22	43%
40	0.51	0.45	50%
50	0.73	0.67	77%
60	0.80	0.75	80%
70	0.93	0.90	57%
80	0.96	0.95	70%
90	0.99	0.99	23%
100	0.99	0.99	23%

The advantage of **TMR** is particularly evident for minefields in which $30 \leq |M| \leq 80$. Figure 3.11 shows that **SP** fails to find minimum-risk paths in 63% of these cases.

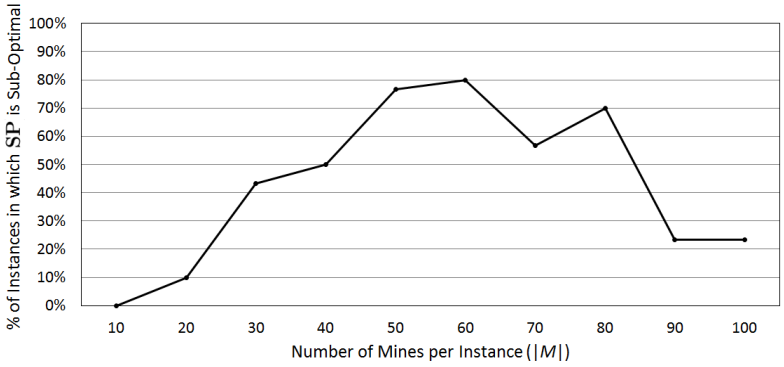


Figure 3.11: Frequency of sub-optimal **SP** solutions. A particularly low or high $|M|$ represents a very sparse or dense minefield, respectively. In these cases, the majority of s-t paths through the minefield have minimal variation in $\mathcal{R}_{E_{st}}$, increasing the chance that **SP** is able to find a minimum-risk path. For moderate densities in which $30 \leq |M| \leq 80$, however, the importance of path selection increases. For a given $|M|$, paths found by **SP** in this range prove to be sub-optimal in up to 80% of our test instances.

Instances outside of this range represent trivially sparse or dense minefields, in which likely survival or failure can be assessed by simple inspection. Inside this range, however, path choice is much more important.

The minimum-risk paths found in G_R by ***MRPA*** possess equal risk to those found by **TMR**. By employing the heuristic ***VGB***, however, ***MRPA*** is able to generate the unique AO Network G_V, implying that the path it finds may not exist in the AO Network G_R used by **TMR**. While we have shown that paths found in G_V may incur unnecessary risk compared to paths in G_R, instances such as the one depicted in Figure 3.12 occur in which ***VGB*** is able to find a path via G_V with a lower $\mathcal{R}_{E_{st}}$ than those found by **TMR** using G_R.

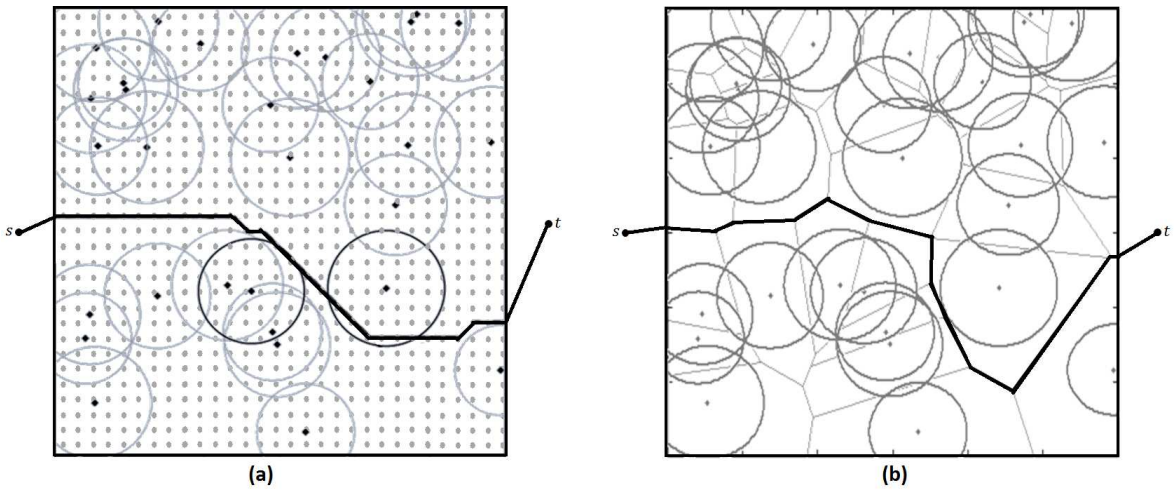


Figure 3.12: (a) A minimum-risk path through an AO with 30 mines found by **TMR** using G_R. (b) Using edges that exist in G_V and not in G_R, ***VGB*** provides an alternative, zero-risk path through the same minefield.

To provide a zero-risk path similar to the one in Figure 3.12(b), G_R would require reduced vertex spacing and/or need to allow retrograde movement. The increase to $|E|$ and $|V|$ associated with these changes greatly reduces the tractability of **TMR**. By contrast, G_V remains comparatively small (see Section 2.1), and Table 3.2 shows that in 66% of our

test cases, minimum-risk paths in the AO Network G_V incur less risk than those found though G_R.

Table 3.2: Columns 2 and 3 compare the average risk of solutions found in G_R by **TMR** to those found in G_V by ***VGB*** across all 30 instances for each density $|M|$. Column 4 indicates the percentage of test cases for each $|M|$ in which the minimum-risk path from G_V incurs less risk than the one found in G_R.

$\|M\|$	$\mathcal{R}_{E_{st}}$		Instances
	G_R	G_V	Improved
10	0	0	0%
20	0.05	0.02	27%
30	0.22	0.18	60%
40	0.45	0.29	73%
50	0.67	0.31	90%
60	0.75	0.57	63%
70	0.90	0.71	80%
80	0.95	0.79	83%
90	0.99	0.86	100%
100	0.99	0.92	90%

3.5.2 Run Times

Edge-additive models such as **SP** rely on shortest-path algorithms that provide an attractive alternative to the theoretical complexity of threat-additive models such as **TMR-1** and **TMR-2**. Table 3.3 shows that the additional time required to identify a threat-additive solution is small, especially when weighed against the improvement to $\mathcal{R}_{E_{st}}$ demonstrated in the previous section.

Average run times predictably increase with minefield density, but remain under five minutes for all instances. Runtimes for **TMR-1** and **TMR-2** initially as much as double those reported in Table 3.3; the following adjustment to their underlying AMPL formulations reduces that: Rather than enforcing the reductions $\boldsymbol{P}$ and $\boldsymbol{R}$ make by restricting constraint indexing, we explicitly eliminate all x_e and y_{jm} variables associated with the edges $\boldsymbol{P}$ and $\boldsymbol{R}$ remove. This requires the bijection $\mathcal{B}_m$ to be redefined to reflect the re-

Table 3.3: Average run times in seconds across all 30 instances of a particular density $|M|$. Both IPs employ a single processing thread and search G_R after reductions $\boldsymbol{P}$ and $\boldsymbol{R}$ have been applied (see Table 3.4 for average values of $|E|$). Times for ***MRPA*** include the time spent by ***VGB*** searching G_V. In each instance, ***VGB*** accounts for fewer than 0.004 seconds.

$\|M\|$	Average Run Time (seconds)			
	SP	**TMR-1**	**TMR-2**	***MRPA*** + ***VGB***
10	0.01	< 0.01	< 0.01	< 0.01
20	0.01	0.03	0.03	0.01
30	0.01	0.14	0.41	5.10
40	0.01	5.78	23.31	0.49
50	0.02	7.97	20.51	0.20
60	0.02	6.20	22.10	0.74
70	0.02	70.75	194.19	0.55
80	0.02	126.09	191.97	0.03
90	0.02	347.23	253.17	0.01
100	0.03	199.58	121.16	0.08

duced E, to maintain the effectiveness of Constraints (3.21) and (3.22), but greatly reduces the size of the final constraint matrix with which the integer programs must contend.

For instances in which $70 \leq |M| \leq 100$, run times for individual instances vary greatly, between one and 4,000 seconds. In these cases, the strength of the initial solution we provide greatly impacts run time. For $|M| \leq 60$, the initial solution, found by $\boldsymbol{H}$ in Figure 3.6, bounds the optimal value of $\mathcal{R}_{Est}$ from above by 0.982 or less. For minefields above this range, the bounds $\boldsymbol{H}$ finds are frequently much weaker. Figure 3.13 provides examples of this phenomenon for **TMR-1**. The density $|M| = 70$ represents an inflection point for run times in Table 3.3, above which the benefits of a stronger LP relaxation, provided by **TMR-2**, begin to outweigh the smaller set of constraints that **TMR-1** possesses. In Figure 3.14, for these highly dense minefields, **TMR-2** consistently provides tighter LP relaxation bounds on the minimum $\mathcal{R}_{E_{st}}$ that are 0.20 or more higher compared to those provided by **TMR-1**.

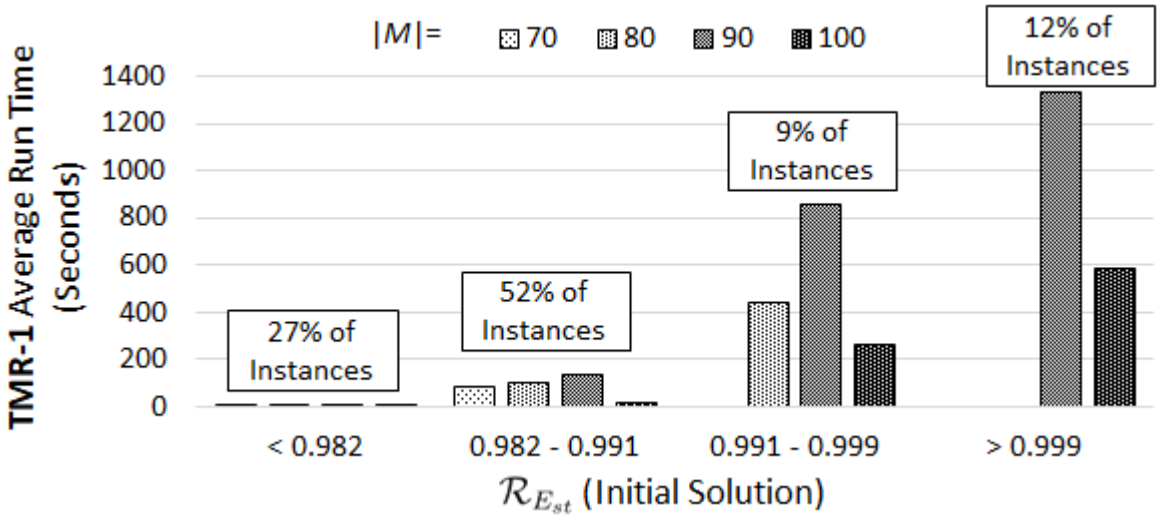


Figure 3.13: Average run times, across instances of high-density minefields, for **TMR-1**. Run times are shown as a function of the objective function value provided by the initial solution found by $\boldsymbol{H}$; higher values of $\mathcal{R}_{Est}$ represent weaker initial upper bounds.

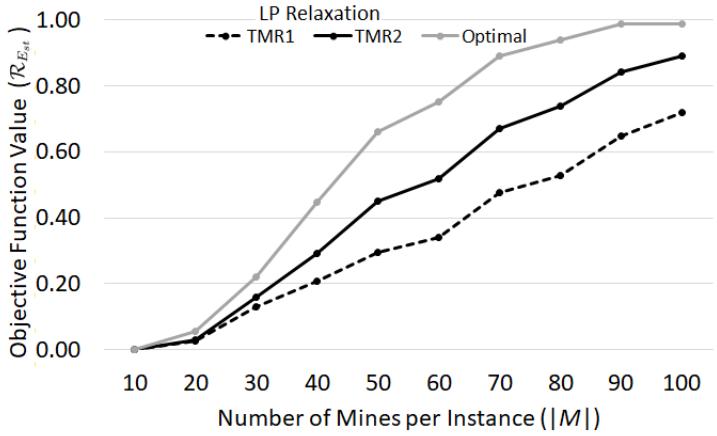


Figure 3.14: Lower bounds on the optimal value of $\mathcal{R}_{Est}$ (shaded line) determined by the LP relaxations in **TMR-1** (dotted line) and **TMR-2** (solid line). As $|M|$ increases, **TMR-2** consistently provides a tighter bound. For all instances in which $|M| = 10$, a zero-risk path is found during the presolve phase.

In addition to helping reduce run times, each bound represented in Figure 3.14 is found in less than one second. Therefore, for realistic navigation scenarios in which a minimum

acceptable $\mathcal{R}_{E_{st}}$ is defined using practical values, e.g., above 0.7, **TMR-2** provides near-instant validation that the minimum-risk path cannot meet these requirements.

Run times for ***MRPA*** benefit from the initial solution provided by ***VGB***. The improved bounds limit the number of paths ***MRPA*** must explore and increase the efficacy of both ***P*** and ***R***. Because ***MRPA*** updates its own bounds during execution whenever it completes an improved s-t path, we modify ***R*** in its application to ***MRPA*** in order to reduce the number of "hops," measured by $|\widehat{E}_{st}|$, required to complete such a path. In addition to eliminating risk-free vertices, which require that *all* incoming and outgoing edges incur zero risk, a single risk-free edge is added to E for *any* consecutive pair of risk-free edges. In practice, this modified ***R*** shortens runtimes by decreasing the number of recursive calls required by **ExtendPath**, but can result in less of a reduction in, or even an overall increase to, $|E|$ due to situations such as the one depicted in Figure 3.15 and in the last entries of Table 3.4.

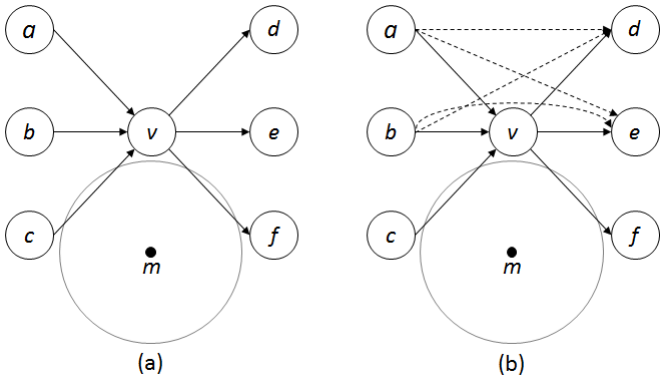


Figure 3.15: A situation in which the modified ***R***, used by ***MRPA***, results in additional edges. In (a), the mine m impinges upon one incoming and one outgoing edge of vertex v; thus, the vertex is not completely risk-free and cannot be removed. In (b), the modified ***R*** connects vertices a and b to vertices d and e with new, direct edges (dashed lines) that bypass vertex v. These four new, risk-free edges, however, do not permit any of their original constituent edges to be removed; edges (a, v) and (b, v) may still be part of paths including (v, f), and edges (v, d) and (v, e) may be part of paths including (c, v). Each dashed edge represents a direct zero-risk edge between vertices whose actual physical path still passes through v.

Table 3.4: Average number of edges, across all instances for each $|M|$, in the AO networks G_V and G_R after reductions $\boldsymbol{P}$ and $\boldsymbol{R}$. The network G_R is originally composed of 2,793 edges, while the size of G_V varies depending on the minefield instance. In the last column, the value of 0 indicates that the bound found by ***VGB*** enables all edges in G_R to be eliminated because the path found in G_V possesses no risk. In the same column, the value of 3,135 indicates the modified $\boldsymbol{R}$ used by ***MRPA*** adds 342 more edges than it removes (see Figure 3.15).

$\|M\|$	$\|E\|$ (After $\boldsymbol{P}$ and $\boldsymbol{R}$ applied)		
	G_V	G_R	
		TMR	***MRPA*** + ***VGB***
10	20	62	0
20	49	405	57
30	78	844	444
40	107	1185	641
50	137	1215	329
60	166	1009	666
70	196	1211	1240
80	225	1174	1457
90	247	1250	1838
100	285	851	3135

As Table 3.4 demonstrates, the effectiveness of the unmodified preprocessing used by **TMR** decreases for values of $|M|$ above 30. Figure 3.16 plots the average number of edges successfully removed from G_R by $\boldsymbol{P}$ and $\boldsymbol{R}$ using the results of $\boldsymbol{H}(G_R, M)$.

For values of $|M|$ larger than 30, the upper bounds on risk, determined by $\boldsymbol{H}$, increase similarly. Both $\boldsymbol{P}$ and $\boldsymbol{R}$ use these bounds to eliminate edges from G_R. Because these bounds increase with minefield density, fewer edges fall within the range identified during preprocessing. If, as an example, the initial path found by $\boldsymbol{H}$ possesses an $\mathcal{R}_{E_{st}}$ of 0.25 for a particular instance with 30 mines, then preprocessing could immediately eliminate each edge that would contribute more than 0.25 to the risk of a path. If, however, $\boldsymbol{H}$ evaluates an instance with 90 mines and can only find a path for which $\mathcal{R}_{E_{st}} = 0.99$, then only edges contributing more than 0.99 risk are immediately eligible for removal.

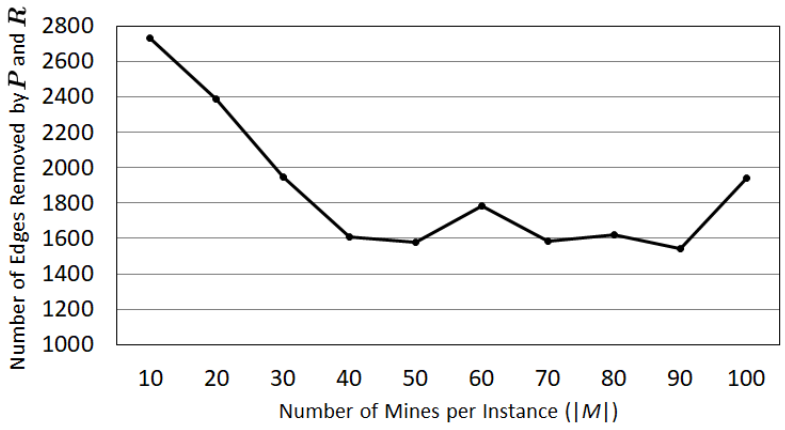


Figure 3.16: Average number of edges removed from G_R by $\boldsymbol{P}$ and $\boldsymbol{R}$ prior to solving **TMR**. Reductions are made using the upper bound on risk associated with the initial path through G_R found by $\boldsymbol{H}$.

3.6 Conclusions

We address the shortcomings faced by traditional edge-additive models when attempting to identify a minimum-risk route for a ship traversing a mapped minefield. Although an edge-additive model solves quickly, it fails to minimize $\mathcal{R}_{E_{st}}$ in 43% of our test cases. We introduce a threat-additive alternative which solves this problem at an acceptable additional computational cost. We implement this approach as both (a) two integer programs (IPs) using commercial optimization software, and (b) a novel implicit path-enumeration algorithm coded in C++ for practical applications in which a non-commercial option is necessary. For the instances we test, CPLEX is able to solve both IP formulations in an average of fewer than 90 seconds. Although our C++ implementation has exponential worst-case complexity, heuristics which generate minefield-specific networks and a good initial solution greatly limit the number of paths that must be explored in practice. In 96% of all test cases, our minimum-risk path algorithm identifies a solution in less time than the integer programs.

Future work will increase the efficiency of our algorithm by improving the bounds on risk and the size of the AO Network. It may be possible to increase $\underline{r}_e$, the lower bound on risk incurred from a single edge, while maintaining the necessary restriction imposed by Equation (3.28). Any such increase will improve all lower-bound calculations along with the initial solution provided by $\boldsymbol{H}$. Furthermore, strengthening $\boldsymbol{H}$ increases the reductions available to $\boldsymbol{P}$ and $\boldsymbol{R}$, and lowers the time spent finding paths. With respect to the AO Network, the high level of success provided by G_V suggests that further research is required to determine if ***VGB*** can be modified such that it guarantees a minimum-risk path while retaining both its $\mathcal{O}(|M|)$ generation complexity and tight bounds on $|V|$ and $|E|$.

Further, while **TMR-2** successfully reduces runtimes, relative to **TMR-1**, for minefields in which $|M| > 80$, the optimal values of $\mathcal{R}_{E_{st}}$ associated with these paths are consistently above 0.99, and do not represent scenarios in which the path would be realistically navigated. As such, the tighter formulation and reduced runtimes provided by **TMR-2** may be more practically applied to military missions in which a large number of threats must be avoided, but in which high values of "risk" are acceptable, e.g., do not directly threaten human lives.

Lastly, the threat-additive model must be updated to account for different mine types. In a 2011 interview about the strategic importance of Guam, a senior U.S. Navy Warfare analyst stated: "We need to be concerned about the Chinese Navy's ability to place covertly small numbers of advanced mines in strategic locations, like the Apra channel..." (Truver 2011). While this underlines the importance of models such as ours, the term "advanced mines" refers to a category of mines which can be triggered by myriad signals and requires that future implementations of our model account for handling detonations as dependent events (LaGrone 2014).

3.7 Acknowledgments

We thank U.S. Navy Admiral (ret) Rick Williams and Capt (ret) James Eagle for their support and interest in real-world applications of ***MRPA***. We appreciate the assistance

provided by Dr. Kevin Wood in the writing of this paper, and would also like to thank Dr. Dinesh Mehta for his advice on and insight into Voronoi diagrams.

3.8 Network Reductions

This appendix provides pseudocode for ***P*** and ***R***, the two network reduction algorithms introduced in Section 3.3.4. Both reductions are called by ***MRPA*** and make the same assumptions that all derived parameters, such as r_e, are stored with G and passed by reference.

Function $\boldsymbol{P}(G, M)\{$
`// Disconnect ‘‘high-risk’’ edges`
foreach $e \in E$ **do**
  if $\left(r_e \geq r_{\widehat{E}_{st}}\right)$ **then** $E \leftarrow E \backslash \{e\}$;
end
`// Remove ‘‘high-risk’’ vertices`
foreach $v \in V \backslash \{s, t\}$ **do**
  $\underline{r}_{E_{sv}} \leftarrow$ risk associated with "shortest" s-v path (via Dijkstra with $\underline{r}_e$);
  if $\underline{r}_{E_{sv}} + \underline{r}_{E_{vt}} \geq r_{\widehat{E}_{st}}$ **then**
    foreach $e \in \overleftarrow{E}_v \cup \overrightarrow{E}_v$ **do** $E \leftarrow E \backslash \{e\}$;
  end
  `// Remove dead ends`
  if $\left(\overleftarrow{E}_v = \emptyset\right)$ **or** $\left(\overrightarrow{E}_v = \emptyset\right)$ **then** $V \leftarrow V \backslash \{v\}$;
end
$\}$

Figure 3.17: Pseudocode for network reduction ***P***. The number of "high-risk" edges and vertices that can be removed depends upon the upper bound $r_{\widehat{E}_{st}}$ set by the initial path $\widehat{E}_{st}$ identified by ***H***.

```
Function R(G, M){
while isReduced = FALSE do
    isReduced ← TRUE;
    foreach v ∈ V\{s, t} do
        // Identify ''risk-free'' vertices
        R_Free ←TRUE;
        foreach e ∈ ←E_v ∪ →E_v do
            foreach m ∈ M_e do
                if |M_e| > 0 then R_Free =FALSE ;
            end
        end
        // If v is risk-free, identify possible reductions
        if R_Free =TRUE then
            isReduced ← FALSE;
            foreach ě = (u, v) ∈ ←E_v do
                foreach ê = (v, u′) ∈ →E_v do
                    // If no shorter edge exists, join edges
                       head-to-tail
                    if (e = (u, u′) ∈ E and (d_ě + d_ê < d_e)) or (e ∉ E) then
                        e ← Join(ě, ê);
                        E ← E ∪ {e}
                    end
                    // Remove incoming edge
                    E ← E\{ě};
                end
                // Remove outgoing edge
                E ← E\{ê};
            end
        end
        // Remove unused vertices
        if (←E_v = ∅) and (→E_v = ∅) then V ← V\{v} ;
    end
end
}
```

Figure 3.18: Pseudocode for $\boldsymbol{R}$. The function loops after each successful reduction until no zero-risk vertices remain. The **Join** function concatenates two edges, head-to-tail, and sets the length and risk of the new edge to the sum of the incoming and outgoing edges' parameters.

CHAPTER 4
NAVIGATING FREE-FLOATING MINEFIELDS VIA TIME-VARYING VORONOI GRAPHS

A paper submitted to the journal *Networks*.

Christopher Richards [12] [13], Dinesh Mehta [14]

4.1 Abstract

Despite their illegality, untethered free-floating naval mines are increasingly employed as part of maritime area-denial operations. By ignoring international treaties, these mines require less effort to acquire and deploy, and result in a far more dynamic-threat environment. In order to assist commanders during breach operation planning, we develop several algorithms that generate a minimum-risk journey through a field of drifting mines. Using Voronoi graphs to capture a series of static snapshots of the operational area, we present a practical methodology for building a fully-connected Time-Varying Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Vertices $\mathcal{V}$ are defined as specific times and locations, and edges $\mathcal{E}$ define the continuous movement of a ship with simple parametric equations. The length, acceleration and risk of each edge is calculated and employed by a threat-additive A^* search to quickly find a plausible, minimum-risk journey through a given minefield within a specific time frame. Using real-world data for a modern-day port and minefields of variable density, we find navigable paths that incur acceptable risk in less than two minutes on average. We also introduce several methods for reducing the size of of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and the time required for its generation.

4.2 Introduction

Free-floating mines are difficult to control and can indiscriminately target military and civilians alike. As a result, their use has been officially prohibited since the Hague Conven-

[12]Major, United States Air Force.
[13]PhD Candidate, Mechanical Engineering Department, Colorado School of Mines.
[14]Professor, Computer Science Department, Colorado School of Mines

tion (Dinstein 2016). According to Changsheng (2005), however, "... international treaties ban the use of drifting sea mines. Of course, actual conditions are really not so ideal." During Operation Desert Storm, Saddam Hussein made extensive use of drift mines to slow coalition forces (Bernitt and Tangredi 2002); in 1984, Colonel Gaddafi used civilian vessels to deploy drifting mines and significantly disrupt traffic through the Suez Canal (Razoux 2015). More recently, U.S. concerns regarding mine capabilities have focused on China and the People's Liberation Army Navy (PLAN). A recent analysis on the PLAN's mine warfare capabilities (Erickson et al. 2009) notes that China has developed and employed drift mines since 1974, and recent Chinese textbooks exult such mines for their ability to be carried by civilian ships and/or laid by hand (Peng and Yucai 2002).

In 2014, then Chairman of the Joint Chiefs of Staff General Martin Dempsey recognized the persistent threat posed by mine warfare and required that Joint Doctrine prepare naval commanders for the presence of armed opposition that is "... characterized by increasingly advanced area denial systems..." (Dempsey 2014). Unfortunately, most U.S. Navy ships cannot assume the mine clearing role of dedicated Mine Countermeasure Vessels (MCMVs). Modern unmanned underwater vehicles (UUVs), however, provide the capability to locate and classify mines (Lin et al. 2008). These UUVs do not yet provide the robust clearance capabilities of an MCMV (Muljowidodo et al. 2009), but are highly effective at determining mine locations (Osborn 2017), measuring environmental parameters such as currents (Shome et al. 2012), and are specifically designed towards small-ship storage and launch (Kongsberg Maritime AS 2011).

In order to assist commanders in evaluating the risk of navigating free-floating minefields, this paper develops algorithms that use the data available from UUV Search-Classify-Map operations to plot a minimum-risk "journey" through such a minefield. These journeys solve the problem, demonstrated in Figure 4.1, of minimum-risk "paths" changing over time as mines experience drift.

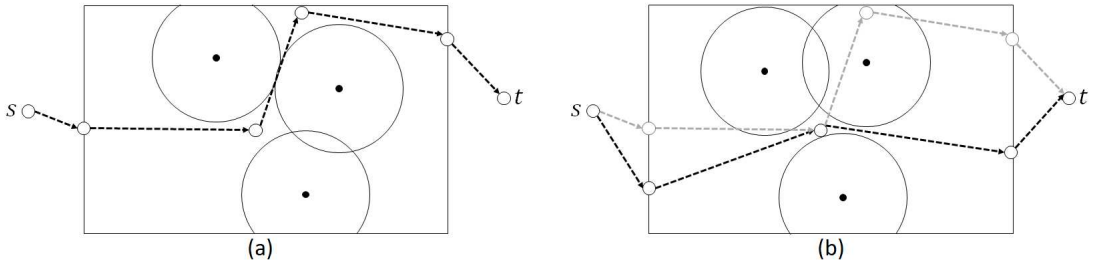


Figure 4.1: Complications inherent to free-floating minefields. In (a), the minimum-risk path (dashed-line) is found using known mine locations. However, one time step later, as depicted in (b), mines have drifted to intersect this path (gray). The previous path now incurs severe, unnecessary risk in comparison to the new, zero-risk path (black) that is available.

The journey we generate includes both a physical path as well as the timings of specific maneuvers. Among potential journeys, approach to each mine is considered as an independent event, risk is minimized according to empirically-derived actuation curves, and physical characteristics of the path are assumed to be restricted by the size and shape of a bounding area of operations.

The remainder of the paper is organized as follows: First, we provide a literature review on minefield navigation, evolving networks, and object drift prediction. We then summarize the criteria that compose the challenge of navigating a drifting minefield and define several of the important terms and parameters. Subsequently, we develop the algorithms necessary to characterize a free-floating minefield, solve practical issues with such a characterization, and efficiently identify a minimum-risk journey. We next discuss the implementation of our our work for a particular real-world port and present results. We conclude with impact and extensions.

4.3 Related Work

Path planning has long been a deep and varied field of study. Early incarnations such as the famous "Seven Bridges of Königsberg" problem (Shields 2012) helped lay the foundations of graph theory in the early 18^{th} century. More recently, advances in robotics

and unmanned-vehicles have led to a wide variety of navigation algorithms, each with their own unique set of objectives (Souissi et al. 2013). Although related to the problem of finding collision-free paths, naval minefield navigation involves several particular challenges. Primary among these are how to accurately measure risk and in what fashion potential paths should be represented.

In the past 200 years, the variety of available sea mines has continued to grow in response to the continual advances made to the ships they target. In contrast to the early mines used by U.S. forces in the 18^{th} and 19^{th} centuries that had very specific operating thresholds and required manual detonation or direct contact (Duncan 1962), modern mines can be placed in almost any littoral environment (i.e., coastal shallows) and can be activated by a myriad signals, including physical contact, sound waves, or electromagnetic signatures (Truver 2011). With specific knowledge about mine type, water depth, and ship characteristics, it is possible to measure with relative accuracy the severity of damage a mine can inflict on a passing ship (Szturomski 2015). In evaluating potential paths, however, researchers tend to focus on determining the likelihood of *avoiding* damage. Boerman (1994) shows that the potential for damage from a sea mine is inversely proportional to R, the linear distance between a ship and a mine. The function that defines the probability of detonation as a function of R is termed the "Actuation Curve" (Department of the Navy 1996), and varies from mine to mine. Although many researchers have constructed viable risk functions based on actuation curves determined through empirical testing (Maldonado 1996, Washburn 2007), these functions do not provide an aggregate risk of passing multiple mines. Rather than aggregate risk, Babel and Zimmermann (2015) uses a risk function to define a minimum "safe" distance around each mine, and then finds paths that avoid these areas while minimizing the length or number of turns within the path. Bekker and Schmid (2006), who represent their paths as a series of connected edges (see below), treat the approach of each edge to each mine as an independent event. In this way, the total risk of a path is defined to be the sum of the risk incurred from each edge in the path. This

aggregation method allows solutions to be found efficiently using shortest-path algorithms like Dijkstra's, but, as Li (2009) points out, can result in "double-counting" risk from mines and ultimately over-estimates the risk of a path. Li mitigates this concern by including and encouraging the use of longer edges, but admits that the possibility of double-counting remains. In order to eliminate any possibility of double-counting, we employ a "threat-additive" model introduced in Chapter 3 by Richards et al. (2017). This formulation precludes the use of Dijkstra's algorithm, but employs an A* search and several heuristics which maintain the tractability of the problem without impact to solution quality, even for instances with extremely dense minefields.

When considering potential paths, many planners intuitively prefer a continuous representation, but Babel and Zimmermann (2015) point out that these methods can only guarantee locally-optimal solutions. Furthermore, these models typically rely on calculus-of-variation techniques and can involve a complex system of differential equations to plot an optimal curvilinear path (Karelahti et al. 2008, Mercer and Sidhu 2007, Zabarankin et al. 2002). Discrete models based on directed networks, on the other hand, can be made arbitrarily close to those of continuous models through purposeful graph construction (Kim and Hespanha 2003, Muhandiramge et al. 2009), and provide global optima. Using a discrete approach, researchers tend to embed a static, directed graph $G(V, E)$ in a representation of the minefield in which vertices V represent waypoints and edges E denote possible segments for ship transit. The challenge of discretizing the minefield in this way is to limit the size of $|E|$ and $|V|$ without eliminating potentially optimal paths. Many researchers (Bekker and Schmid 2006, Boerman 1994, Li 2009) generate a regular grid by evenly distributing vertices V, with between four and eight edges directed outward toward neighboring vertices. In these constructions, $|V|$ and $|E|$ depend largely on the area a minefield spans and can greatly affect the computational complexity depending on what level of granularity is acceptable. Furthermore, Babel and Zimmermann (2015) argue that paths found in such networks may be not be navigable by a ship, and focus on developing a network with edges

that better reflect the maneuvers available to seagoing vessels. Li (2009) and Richards et al. (2017) eliminate difficult, acute turns by restricting retrograde movement and introducing distance penalties that avoid unnecessary maneuvers. For a typical minefield composed of a single mine type, Richards et al. (2017) show that a Voronoi graph generated around the locations of the mines M in a minefield, provides paths that maximize the minimum distance to each mine, limits both $|V|$ and $|E|$ to $\mathcal{O}(|M|)$, and maximizes the minimum angle between edges.

For the free-floating minefields we study, perhaps the most closely applicable networks are the Zermelo-Voronoi diagrams (ZVD) introduced very recently by Sun and Tsiotras (2017). Unlike a normal Voronoi diagram, a ZVD assumes that its generating points are under the influence of drift. Like many other applications of Voronoi diagrams generated around moving points (Bakolas and Tsiotras 2010, Devillers et al. 1996), however, Sun and Tsiotras (2017) considers the moving points as agents and is primarily concerned with the *partitions* the ZVD provides as a means to decide which agent should pursue a given target. In contrast, we are interested in the *edges* that define the diagram as a means to construct a network within which possible ship paths can be identified that maximize the minimum distance to mines. This distinction is important. Rather than constructing a series of ZVD's as "snapshots" and making a decision at each time step, we must generate a single Time Varying Graph (TVG).

Casteigts et al. (2012) provides a robust survey and useful taxonomy of recent TVG structures and methods. Using what Casteigts terms "graph-centric evolution," a discrete TVG can be constructed by considering a "sequence of footprints" in which a series of static graphs from each time step capture the possible connections between vertices in subsequent time steps. While this approach has been used effectively to model information and/or data flow through social- and sensor-networks (Tang et al. 2010, Ferreira 2004), much less research exists concerning how to construct a TVG that represents the *physical* movement between two static and temporally adjacent graphs.

In comparison to previous research concerning naval-minefield navigation, we provide several unique contributions. First, we consider free-floating, rather than static, minefields. We employ empirically-gathered environmental data along with ocean-drift models to predict the location of mines at each future time step (see Section 4.1). Second, while previous models find a minimum-risk physical "path" through a minefield, we construct time-dependent "journeys" that account for mine movement. Letting a series of static Voronoi graphs serve as our "sequence of footprints," we build a discrete TVG with parametric equations that continuously define ship location, acceleration, and risk. Lastly, prior research identifies topological changes that can occur within a Voronoi-based TVG, and defines limits for their frequency. We introduce a practical approach to handling these changes that ensures the connectivity of our TVG and that a feasible journey exists.

4.4 Preliminaries and Notation

This section formally defines important military and mathematical concepts and introduces basic notation.

4.4.1 Free-Floating Minefield

We define a Free-Floating Minefield (FFM), illustrated in Figure 4.2, as a collection of mines $m \in M$ that exist in a body of water with physical boundaries approximated by a polygonal Area of Operations (AO).

The corners of the AO are defined in a clockwise manner by an ordered set of points $\rho_b,\ b = 1, \ldots, B$ such that $\rho_1 = \rho_B$. Subsets $\mathcal{S}$ and $\mathcal{T}$ are comprised of consecutive ρ_b, and define potential entry and exit to the AO, respectively. The FFM is embedded into an $x-y$ plane and is assumed to be navigated in a maximum of T timesteps. Thus, the coordinates of each mine m at time step τ can be represented as (m_x^τ, m_y^τ). Current and windspeed at position (x, y) for time steps $\tau = 1, \ldots, T$ are represented via the functions $\chi_C^\tau(x, y)$ and $\chi_W^\tau(x, y)$, respectively. These functions, in turn, are used to define the function $P^\tau(m)$, which returns the predicted position of mine m at time $\tau + 1$ as a function of its position at time τ. (These functions are derived and defined in Section 4.1.)

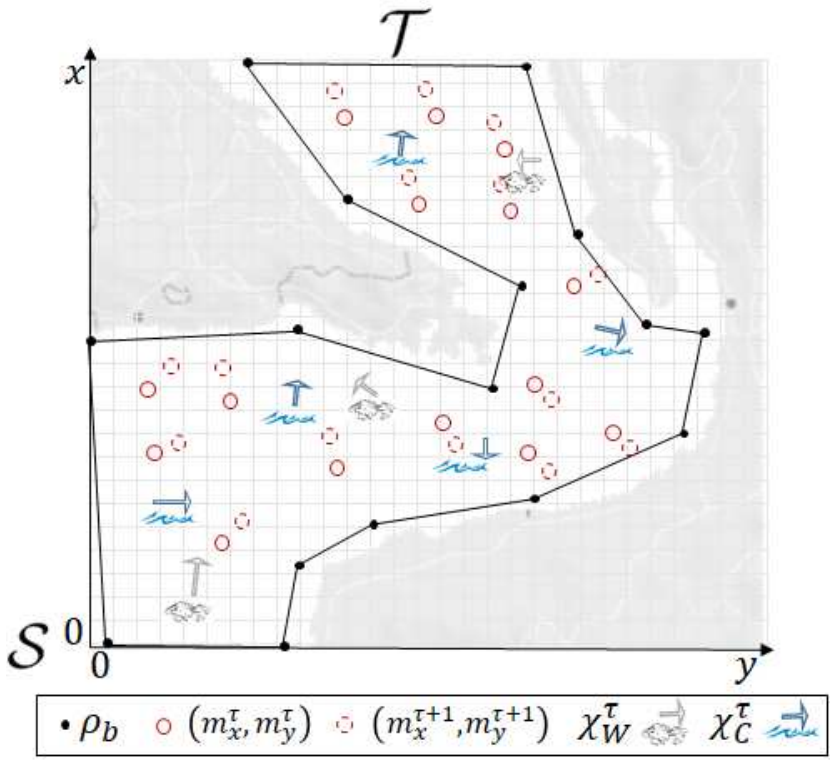


Figure 4.2: A notional free-floating minefield, annotated with parameters defined in this section. In this example, $B = 17$, $\rho_1 = \rho_B = (0, 0)$, $\mathcal{S} = \{\rho_{17}, \rho_1, \rho_2\}$, $\mathcal{T} = \{\rho_{10}, \rho_{11}\}$ and, while not explicitly depicted on the map, the vectors χ_C^τ and χ_W^τ are continuously defined at every point (x, y).

4.4.2 Static and Time-Varying Network Representations of the AO

In addition to the mines M that populate the FFM, we construct a set of "dummy" mines M_β. These mines are positioned outside of the FFM at a distance δ orthogonal to each vector $\overrightarrow{\rho_b, \rho_{b+1}}$, at regular intervals of δ. At each time step τ we define $G_\tau(V_\tau, E_\tau)$ to be the Voronoi diagram generated using the locations of $M \cup M_\beta$ at time τ. In two dimensions, this partitions the AO into cells such that all points within a given cell are closer to the mine it contains than they are to any other mine (Edelsbrunner 2014), as demonstrated in Figure 4.3.

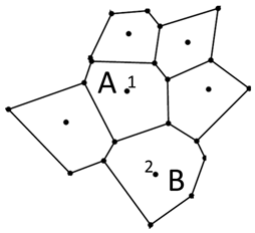


Figure 4.3: Voronoi diagram cell construction. All points in cell A are closer to mine 1 than they are to any other mine; the same is true for the points in cell B and mine 2. Every point along the edge between cells A and B is equidistant to mines 1 and 2.

The inclusion of M_β ensures that G_τ sufficiently models the boundary of the AO, while M ensures that each edge $e \in E_\tau$ maximizes the minimum distance to any mine $m \in M$ at time τ. (See Section 4.2 for specifics concerning the choice of δ and the generation of M_β.) A source vertex s is connected to each edge separating pairs of dummy mines associated with $\mathcal{S}$, while each edge separating pairs of dummy mines associated with $\mathcal{T}$ is connected to a sink vertex t. Remaining edges between dummy mines are removed, and all other edges are assumed to be bi-directional.

We assume that each G_τ is not degenerate, i.e., that the in- and out-degree of each vertex $v \in V_\tau$ is exactly three. Using this fact, we identify each vertex $v \in V_\tau$ with a unique triplet such that $v = (a, b, c)$, where a, b, and c are the three mines adjacent to v. Furthermore, we define the coordinates of v at time τ as (v_x^τ, v_y^τ). Although these coordinates vary for each time step, the set of triplets that represent $v \in V_\tau$ remain unchanged for $V_{\tau+1}$ unless the Voronoi diagram becomes degenerate at some point between time τ and $\tau + 1$. (See Section 4.3.2 for explanation and strategies concerning these topological events.) We define $\mathcal{P}(V_\tau)$ to be the set of $v \in V_\tau$ that are unchanged by topological events between time τ to $\tau + 1$, i.e., $V_\tau \cap V_{\tau+1} = \mathcal{P}(V_\tau)$.

The TVG $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is constructed such that each $\vartheta \in \mathcal{V}$ is defined as (v, τ) and denotes a static vertex $v \in V_\tau$ at a specific time τ with position (v_x^τ, v_y^τ). Edges $\varepsilon \in \mathcal{E}$ are defined as (ϑ, ϑ'), and represent possible segments of ship travel. Each edge is defined between

eligible vertices in $\mathcal{V}$ by continuous parametric equations $x_\varepsilon(\hat{\tau})$ and $y_\varepsilon(\hat{\tau})$ over the domain $0 \leq \hat{\tau} \leq 1$. These equations are defined in Section 4.3.1, and Figure 4.4 illustrates an instance of ε.

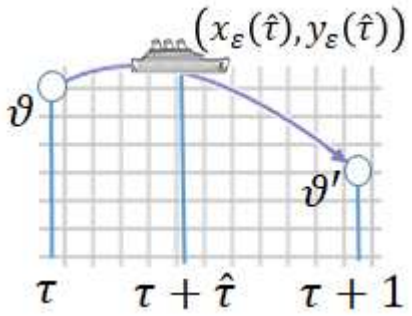


Figure 4.4: Ship traveling along edge $\varepsilon = (\vartheta, \vartheta')$. The edge begins at timestep τ and ends at the subsequent timestep $\tau + 1$. Ship position along ε at any time $\tau + \hat{\tau}$ between τ and $\tau + 1$ is defined by the parametric equations $x_\varepsilon(\hat{\tau})$ and $y_\varepsilon(\hat{\tau})$.

A journey $\mathcal{E}_{st}^{\tau\tau'}$ in $\mathcal{G}$ is a simple, connected subsequence of edges in $\mathcal{E}$ that begins at vertex $(s, \tau) \in \mathcal{V}$ for some $\tau \geq 1$, and ends at vertex $(t, \tau') \in \mathcal{V}$ such that $\tau \leq \tau' \leq T$.

4.4.3 Risk, Acceleration, and Length

For each $\varepsilon \in \mathcal{E}$, the risk a ship incurs from mine m by traveling along ε is defined based on the closest point of approach ε makes to m. We call the distance at which this occurs $c_{\varepsilon m}$, and let $p_{\varepsilon m}$ be the probability of mine m detonating as a function of $c_{\varepsilon m}$. The risk to a ship traversing edge ε from mine m is set to $r_{\varepsilon m}$ using a logarithmic transform such that

$$r_{\varepsilon m} \equiv -\log(1 - p_{\varepsilon m}). \tag{4.1}$$

The measure in Equation (4.1) is used as a positive "weighting" for each edge $\varepsilon \in \mathcal{E}$, and allows independently-accrued risk to be aggregated additively. Using the threat-additive model developed by Richards et al. (2017), the total risk of a journey $\mathcal{E}_{st}^{\tau\tau'}$ is denoted as $\mathcal{R}_{\mathcal{E}_{st}^{\tau\tau'}}$ and is defined by

$$\mathcal{R}_{\mathcal{E}_{st}^{\tau\tau'}} \equiv \sum_{m \in M} \max_{\varepsilon \in \mathcal{E}_{st}^{\tau\tau'}} r_{\varepsilon m}. \tag{4.2}$$

In addition to the risk associated with a journey, we define its physical length as $\mathcal{D}_{\mathcal{E}_{st}^{\tau\tau'}}$. Letting d_ε denote the length of individual edges, we have

$$\mathcal{D}_{\mathcal{E}_{st}^{\tau\tau'}} = \sum_{\varepsilon \in \mathcal{E}_{st}} d_\varepsilon. \tag{4.3}$$

Lastly, a ship traveling along an edge ε must accelerate or decelerate depending on its physical characteristics. Due to the way in which edges in $\mathcal{E}$ are defined (see Section 4.3), the acceleration along each edge is constant, and defined to be α_ε.

4.5 Preprocessing and Algorithms

The following section explicitly defines pseudocode for the algorithms we develop to characterize a given FFM and find the minimum-risk journey through $\mathcal{G}$. All parameters are assumed to be passed by reference, and use the notation developed in the previous section.

4.5.1 Mine Locations

Generating the future location of free-floating mines requires three key steps: (*i*) Determination of the initial locations of mines, (*ii*) Collection of environmental drift data from the AO, and (*iii*) Construction of a mathematical model for predicting mine movement. Both (*i*) and (*ii*) can be obtained via UUVs. Through Search-Classify-Map operations, these drones have proven highly effective at identifying mine locations (Osborn 2017). Furthermore, many UUVs also possess the ability to measure environmental parameters including current and windspeed (Shome et al. 2012). The wireless connectivity of such UUVs means that allowing several to linger within an AO after mapping has been completed grants real-time access to the parameters required by (*ii*). As to (*iii*), Breivik and Allen (2008) have done a great deal of work to develop search-and-rescue models that predict the locations of free-floating items at sea. They find that the drift or "leeway" of an object is a function of current and the "slippage" caused by wind due to an object's size

and physical characteristics. The slippage, in turn, is divided into downwind and crosswind components, the former parallel to the wind vector, the latter orthogonal. Data collected by UUVs define $\chi_C^\tau(x, y)$ and $\chi_W^\tau(x, y)$, the current and wind velocity vectors, respectively, at time step τ. Letting $\widehat{\chi}_W^\tau$ be the vector orthogonal to the wind vector, the position of mine m at time step $\tau + 1$ is found by the function P^τ, defined as

$$P^\tau(m) \equiv (m_x^\tau, m_y^\tau) + w_d \cdot \chi_W^\tau(m_x^\tau, m_y^\tau) + w_c \cdot \widehat{\chi}_W^\tau(m_x^\tau, m_y^\tau) + \chi_C^\tau(m_x^\tau, m_y^\tau), \tag{4.4}$$

where w_d, and w_c are parameters defined by the object's physical characteristics. We use values of w_d and w_c derived by Breivik et al. (2011) during empirical tests on the leeway characteristics of a naval mine. With P^τ defined for each time step τ, locations for each mine $m \in M$ are generated by Algorithm 1.

Function **GenFields**$\left(M, \bigcup_{\tau=1\ldots T} P^\tau\right)\{$
`// Predict mine locations at each time step` τ
for $\tau = 2, \ldots, T$ **do**
  `// Create new set of mine locations`
  foreach $m \in M$ **do**
    $(m_x^\tau, m_y^\tau) \leftarrow P^{\tau-1}(m)$;
  end
end
}

Algorithm 1: Pseudocode to populate M with mine locations at each time step τ. The algorithm assumes that initial mine positions, i.e., $(m_x^1, m_y^1)\ \forall m \in M$ are set according to gathered UUV data.

4.5.2 Static Voronoi Graphs

Once M has been updated to include the locations of each mine at future time steps, the static graph $G_\tau(V_\tau, E_\tau)$ must be generated for each time step τ. Generation of G_τ is done in three steps: (i) Determine and/or include the locations of dummy mines M_β, (ii)

Generate a Voronoi diagram, and (*iii*) Complete G_τ by connecting source and sink vertices s and t, and removing unused edges.

Dummy mine locations remain constant and are determined once using $\overrightarrow{\rho_b\rho_{b+1}}$ for $b = 1 \dots B-1$, the edges that define the polygonal boundary of the AO. The vector $\overrightarrow{\delta}_b$ is defined with length δ, is orthogonal to $\overrightarrow{\rho_b\rho_{b+1}}$, and directed away from the interior of the FFM. In this way, the location of each mine $\bar{m} \in M_\beta$, as shown in Figure 4.5, is calculated as

$$(\bar{m}_x, \bar{m}_y) \equiv \frac{n\delta}{||\overrightarrow{\rho_b\rho_{b+1}}||} \cdot \overrightarrow{\rho_b\rho_{b+1}} + \overrightarrow{\delta}_b, \forall n \in \mathbb{N}_0 : n\delta \leq ||\overrightarrow{\rho_b\rho_{b+1}}||. \tag{4.5}$$

Because dummy mines M_β represent physical boundaries and are unaffected by drift, their coordinates in Equation (4.5) remain constant and, thus, have no associated τ superscript. The parameter δ is set to the smallest possible value for which the edges in E_τ not separating mines in M_β cease to intersect the AO boundary. Letting the function **Voronoi** $(\{(x_i, y_i) : i = 1 \dots N\})$ return the Voronoi diagram around N points, at each time step τ, we set

$$G_\tau(V_\tau, E_\tau) \equiv \textbf{Voronoi}\left(\bigcup_{\bar{m} \in M_\beta} (\bar{m}_x, \bar{m}_y) \cup \bigcup_{m \in M} (m_x^\tau, m_y^\tau)\right). \tag{4.6}$$

Lastly, as illustrated in Figure 4.5(b), we remove each edge in E_τ that extends between mines $\bar{m} \in M_\beta$ to "infinity", and proceed to connect source and sink vertices s and t.

Letting $M_\mathcal{S}$ and $M_\mathcal{T}$ represent the subset of dummy mines associated with entry and exit boundaries $\mathcal{S}$ and $\mathcal{T}$, we add edges $e = (s, v)$ and $e = (v, t)$ to E_τ to represent possible entry and egress to G_τ, respectively, by ensuring that

$$(s, v) \in E_\tau \ \forall v = (a, b, c) \in V_\tau : |\{a, b, c\} \cap M_\mathcal{S}| = 2, \tag{4.7}$$

$$(v, t) \in E_\tau \ \forall v = (a, b, c) \in V_\tau : |\{a, b, c\} \cap M_\mathcal{T}| = 2. \tag{4.8}$$

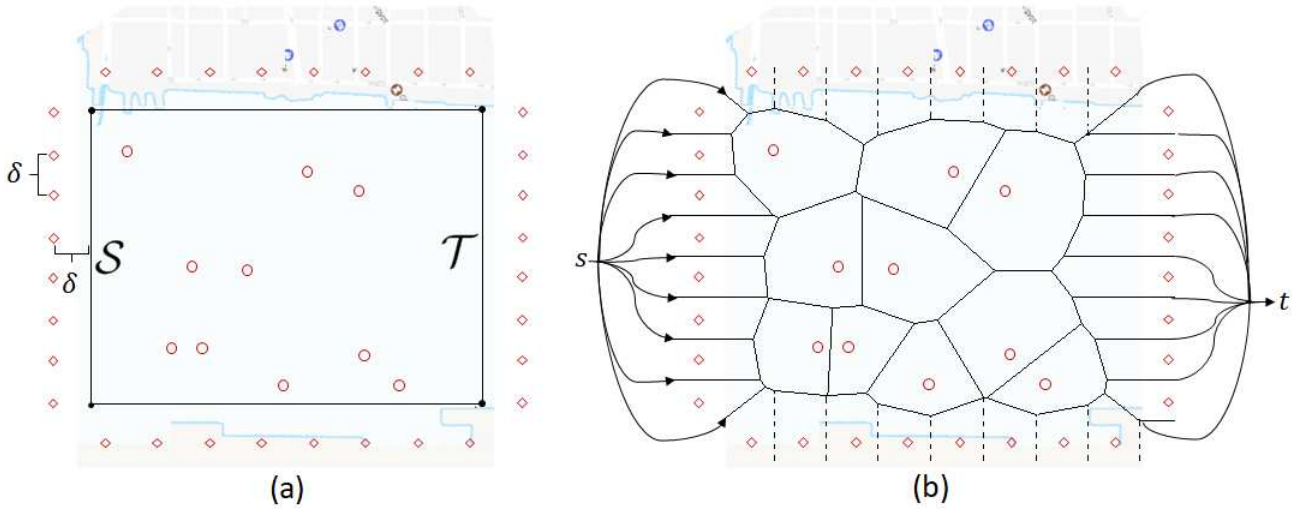


Figure 4.5: (a) Simple FFM at a particular time τ. Mines $m \in M$ are represented by red circles, while dummy mines M_β are denoted by red diamonds outside the rectangular AO. (b) Static graph G_τ generated with Voronoi diagram about $M \cup M_\beta$. Arrows indicate directed edges out of and into source and sink vertices s and t, respectively. Dashed edges are discarded after initial Voronoi construction.

With the parameters and procedures associated with (i)-(iii) defined in these ways, $G_\tau(V_\tau, E_\tau)$ can be generated at each time step τ by Algorithm 2.

5.3 Time Varying Graph ($\mathcal{G}(\mathcal{V}, \mathcal{E})$)

Once the static graph G_τ has been created for each time step τ, we create the TVG $\mathcal{G}(\mathcal{V}, \mathcal{E})$. While each G_τ is a "snapshot" of the paths through the FFM at a particular time step, $\mathcal{G}$ defines movement from a point v to another point v' "between" consecutive time steps. As such, we define $\mathcal{G}$ according to two cases: (i) Under the assumption that $\{v, v'\} \in \mathcal{P}(V_\tau)$, i.e., topological changes between G_τ and $G_{\tau+1}$ do not affect where movement begins and ends, and (ii) $v \notin \mathcal{P}^\tau$ or $v' \notin \mathcal{P}^\tau$, i.e., one or more of the endpoints disappear between time steps τ and $\tau + 1$.

Function **GenStatics**$\left(M, \bigcup_{b=1\ldots B} \rho_b, \overrightarrow{\delta}_b, \mathcal{S}, \mathcal{T}\right)$ {

```
// Generate bounding dummy mine set
```

$M_\beta = \emptyset$;
$M_{\mathcal{S}} \leftarrow \emptyset$;
$M_{\mathcal{T}} \leftarrow \emptyset$;
for $b = 1 \ldots B-1$ **do**
  $\overrightarrow{\mathbf{L}} \leftarrow \frac{\delta \cdot \overrightarrow{\rho_b \rho_{b+1}}}{||\overrightarrow{\rho_b \rho_{b+1}}||}$;
  while $||\overrightarrow{L}|| \leq ||\overrightarrow{\rho_b \rho_{b+1}}||$ **do**
    NewMine$(\bar{m})$;
    $(\bar{m}_x, \bar{m}_y) \leftarrow \overrightarrow{L} + \overrightarrow{\delta_b}$;
    $M_\beta \leftarrow M_\beta \cup \{\bar{m}\}$;
    if $\rho_b \in \mathcal{S}$ **then** $M_{\mathcal{S}} \leftarrow M_{\mathcal{S}} \cup \{\bar{m}\}$;
    if $\rho_b \in \mathcal{T}$ **then** $M_{\mathcal{T}} \leftarrow M_{\mathcal{T}} \cup \{\bar{m}\}$;
    $\overrightarrow{\mathbf{L}} \leftarrow \overrightarrow{\mathbf{L}} + \frac{\delta \cdot \overrightarrow{\rho_b \rho_{b+1}}}{||\overrightarrow{\rho_b \rho_{b+1}}||}$;
  end
end

```
// Build static graphs
```

for $\tau = 1 \ldots T$ **do**
  $G_\tau(V_\tau, E_\tau) \leftarrow$ **Voronoi** $\left(\bigcup_{\bar{m} \in M_\beta} (\bar{m}_x, \bar{m}_y) \cup \bigcup_{m \in M} (m_x^\tau, m_y^\tau)\right)$;

```
  // Remove dummy edges
```

  foreach $e \in E_\tau$ **s.t.** $e = (v, \infty)$ **or** $e = (\infty, v)$ **do**
    $E_\tau \leftarrow E_\tau \setminus \{e\}$;
  end

```
  // Create and connect source and sink nodes
```

  $V_\tau \leftarrow V_\tau \cup \{s, t\}$;
  foreach $v = (a, b, c) \in V_\tau$ **do**
    if $|\{a, b, c\} \cap M_{\mathcal{S}}| = 2$ **then** $E_\tau \leftarrow E_\tau \cup \{(s, v)\}$;
    if $|\{a, b, c\} \cap M_{\mathcal{T}}| = 2$ **then** $E_\tau \leftarrow E_\tau \cup \{(v, t)\}$;
  end
end
}

Algorithm 2: Pseudocode to generate static graphs G_τ.

5.3.1 Case 1: No Topological Change

Each vertex $\vartheta \in \mathcal{V}$ represents a static vertex v as it "exists" at a time step τ such that

$$\vartheta = (v, \tau)\ \forall v \in V_\tau, \tau = 1 \dots T. \tag{4.9}$$

Likewise, for each edge $e = (v, v') \in E_\tau, \tau = 1 \dots T$, if $v, v' \in \mathcal{P}(V_\tau)$, we include edge $\varepsilon = (\vartheta, \vartheta') \in \mathcal{E}$ such that $\vartheta = (v, \tau)$ and $\vartheta' = (v', \tau + 1)$. Figure 4.6 builds on Figure 4.4 and illustrates how each edge ε models the motion of a ship traveling from ϑ to ϑ'.

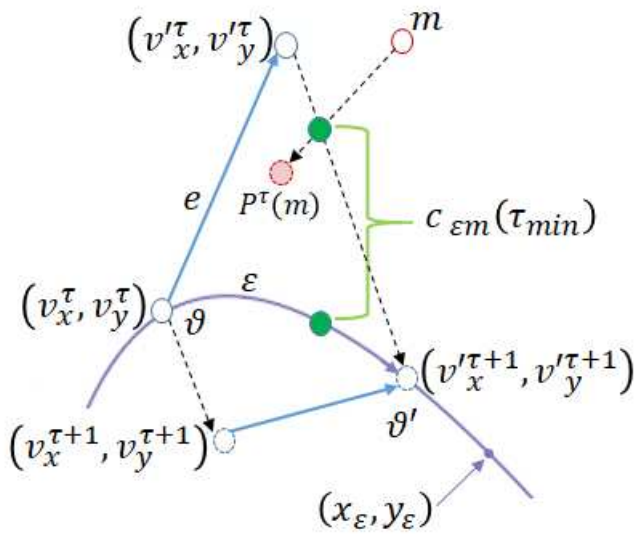


Figure 4.6: Parabolic parametric path taken by ε traveling from $\vartheta = (v, \tau)$ to $\vartheta = (v', \tau+1)$. The path of the edge models movement along the static edge $e = (v, v')$ as it transitions from time step τ to $\tau + 1$.

Because the points v and v' and, thus, the minimum-risk edge e connecting them move between time τ and $\tau + 1$, accurately describing ship movement along ε must be done parametrically. To do so, first we define the linear path of a vertex v from (v^{τ}_x, v^{τ}_y) to $(v^{\tau+1}_x, v^{\tau+1}_y)$ as

$$x_v(\hat{\tau}) \equiv \left(v_x^{\tau+1} - v_x^{\tau}\right)\hat{\tau} + v_x^{\tau}, \tag{4.10}$$

$$y_v(\hat{\tau}) \equiv \left(v_y^{\tau+1} - v_y^{\tau}\right)\hat{\tau} + v_y^{\tau}, \tag{4.11}$$

where $0 \leq \hat{\tau} \leq 1$ and represents the fraction of time step $\tau + 1$ that has passed. In order to describe the path ε follows from $\vartheta = (v_x^{\tau}, v_y^{\tau})$ to $\vartheta' = (v'^{\tau}_x, v'^{\tau}_y)$, we develop equations similar to (4.10) and (4.11), but account for the movement of v and v' by representing their coordinates paremetrically.

$$x_{\varepsilon}(\hat{\tau}) \equiv \left(x_{v'}(\hat{\tau}) - x_v(\hat{\tau})\right)\hat{\tau} + x_v(\hat{\tau}), \tag{4.12}$$

$$y_{\varepsilon}(\hat{\tau}) \equiv \left(y_{v'}(\hat{\tau}) - y_v(\hat{\tau})\right)\hat{\tau} + y_v(\hat{\tau}). \tag{4.13}$$

Expanding Equations (4.12) and (4.13), we obtain the quadratic parametric equations

$$x_{\varepsilon}(\hat{\tau}) = \left(\Delta x_{v'} - \Delta x_v\right)\hat{\tau}^2 + \left(v_x'^{\tau} + v_x^{\tau+1} - 2v_x^{\tau}\right)\hat{\tau} + v_x^{\tau}, \tag{4.14}$$

$$y_{\varepsilon}(\hat{\tau}) = \left(\Delta y_{v'} - \Delta y_v\right)\hat{\tau}^2 + \left(v_y'^{\tau} + v_y^{\tau+1} - 2v_y^{\tau}\right)\hat{\tau} + v_y^{\tau}, \tag{4.15}$$

where

$$\Delta[x|y]_v \equiv v_{[x|y]}^{\tau+1} - v_{[x|y]}^{\tau}, \tag{4.16}$$

$$\tau \in \mathbb{R} : \ 0 \leq \hat{\tau} \leq 1. \tag{4.17}$$

With ε defined in terms of parametric equations x_{ε} and y_{ε}, it is possible to calculate the parameter d_{ε}. The arc length of a parametric curve can be derived through integration and knowledge of the parametric equations' derivatives with respect to t (Stewart 2015). Thus, we define linear derivatives

$$\frac{d}{d\hat{\tau}} x_{\varepsilon}(\hat{\tau}) = 2\left(\Delta x_{v'} - \Delta x_v\right)\hat{\tau} + v_x'^{\tau} + v_x^{\tau+1} - 2v_x^{\tau}, \tag{4.18}$$

$$\frac{d}{d\hat{\tau}} y_\varepsilon(\hat{\tau}) = 2\left(\Delta y_{v'} - \Delta y_v\right)\hat{\tau} + v_y'^{\tau} + v_y^{\tau+1} - 2v_y^{\tau}. \tag{4.19}$$

Using these equations, the length d_ε of edge ε is found by integrating over $\hat{\tau}$ from 0 to 1 according to

$$d_\varepsilon \equiv \int_0^1 \sqrt{\left(\frac{d}{d\hat{\tau}} x_\varepsilon(\hat{\tau})\right)^2 + \left(\frac{d}{d\hat{\tau}} y_\varepsilon(\hat{\tau})\right)^2}\, d\hat{\tau}. \tag{4.20}$$

Similarly, the acceleration of a particle moving according to parametric equations is described according to the vector $\left(\frac{d^2}{d\hat{\tau}^2} x_\varepsilon(\hat{\tau}), \frac{d^2}{d\hat{\tau}^2} y_\varepsilon(\hat{\tau})\right)$, calculated using the equations' second derivatives (Stewart 2015). Thus, the parameter α_ε, the magnitude of a ship's acceleration traversing edge ε, can be calculated with the derivatives of Equations (4.18) and (4.19) . As mentioned in Section 3, these derivatives are constant, taking the forms

$$\frac{d^2}{d\hat{\tau}^2} x_\varepsilon(\hat{\tau}) = 2\Delta x_{v'} - 2\Delta x_v, \tag{4.21}$$

$$\frac{d^2}{d\hat{\tau}^2} y_\varepsilon(\hat{\tau}) = 2\Delta y_{v'} - 2\Delta y_v, \tag{4.22}$$

and allow the parameter α_ε to be defined as

$$\alpha_\varepsilon \equiv \sqrt{\left(\frac{d^2}{d\hat{\tau}^2} x_\varepsilon(\hat{\tau})\right)^2 + \left(\frac{d^2}{d\hat{\tau}^2} y_\varepsilon(\hat{\tau})\right)^2}. \tag{4.23}$$

In order to obtain the "risk" $r_{\varepsilon m}$, defined in Equation (4.1), we must first define the parametric equations that govern the linear path of a mine m:

$$x_m(\hat{\tau}) \equiv \left(m_x^{\tau+1} - m_x^{\tau}\right)\hat{\tau} + m_x^{\tau}, \tag{4.24}$$

$$y_m(\hat{\tau}) \equiv \left(m_y^{\tau+1} - m_y^{\tau}\right)\hat{\tau} + m_y^{\tau}. \tag{4.25}$$

Next, we define the distance between a point moving according to Equations (4.12) and (4.13), and a mine m moving according to Equations (4.24) and (4.25), as a function of $\hat{\tau}$:

$$c_{\varepsilon m}(\hat{\tau}) \equiv \sqrt{\left(x_m(\hat{\tau}) - x_\varepsilon(\hat{\tau})\right)^2 + \left(y_m(\hat{\tau}) - y_\varepsilon(\hat{\tau})\right)^2}. \tag{4.26}$$

Equation (4.26) is simply the Euclidean norm with x- and y-coordinates substituted with the appropriate parametric positions. Setting the derivative of the radicand in Equation (4.26) to zero and solving for $\hat{\tau}$ gives $\hat{\tau}_{min}$; the unique value of $\hat{\tau}$ at which the distance between the paths of the mine m and the path traced by Equations (4.12) and (4.13) is smallest. Because the paths of a ship traversing edge ε begins at a timestep τ and ends at timestep $\tau + 1$, the minimum distance between a ship traversing ε and a mine m is defined piecewise as

$$c_{\varepsilon m} \equiv \begin{cases} c_{\varepsilon m}(0) & \text{if } \hat{\tau}_{min} < 0, \\ c_{\varepsilon m}(\hat{\tau}_{min}) & \text{if } 0 \leq \hat{\tau}_{min} \leq 1, \\ c_{\varepsilon m}(1) & \text{if } \hat{\tau}_{min} > 1. \end{cases} \tag{4.27}$$

Using results from Equation (4.27), we let

$$p_{\varepsilon m} \equiv W(c_{\varepsilon m}) \tag{4.28}$$

where W denotes the Washburn actuation function as derived by Maldonado (1996), and calculate $r_{\varepsilon m}$ according to Equation (4.1). The function **NewEdge**(e, e', τ, M) in Algorithm 3 summarizes the steps to create each TVG edge ε and, for instances in which ε begins at

s or ends at t, sets $d_\varepsilon = \alpha_\varepsilon = r_{\varepsilon m} = 0$, as these edges represent a choice of entry and exit vertex, respectively, and do not represent actual travel.

5.3.2 Case 2: Topological Change Occurs

As alluded to in Section 3, it is possible that $\mathcal{P}(V_\tau) \neq V_\tau$ due to topological changes between time steps τ and $\tau + 1$. Albers et al. (1998) proves that, as the points defining a Voronoi diagram move, the resultant graph can experience only two types of changes: (i) The appearance or disappearance of a point on the interior of the graph, or (ii) The appearance or disappearance of a point on the convex hull of the graph. Because each instance of G_τ discards the perimeter edges generated due to dummy mines in M_β, case (ii) does not apply to our research. Case (i), on the other hand, results in the situation illustrated in Figure 4.7:

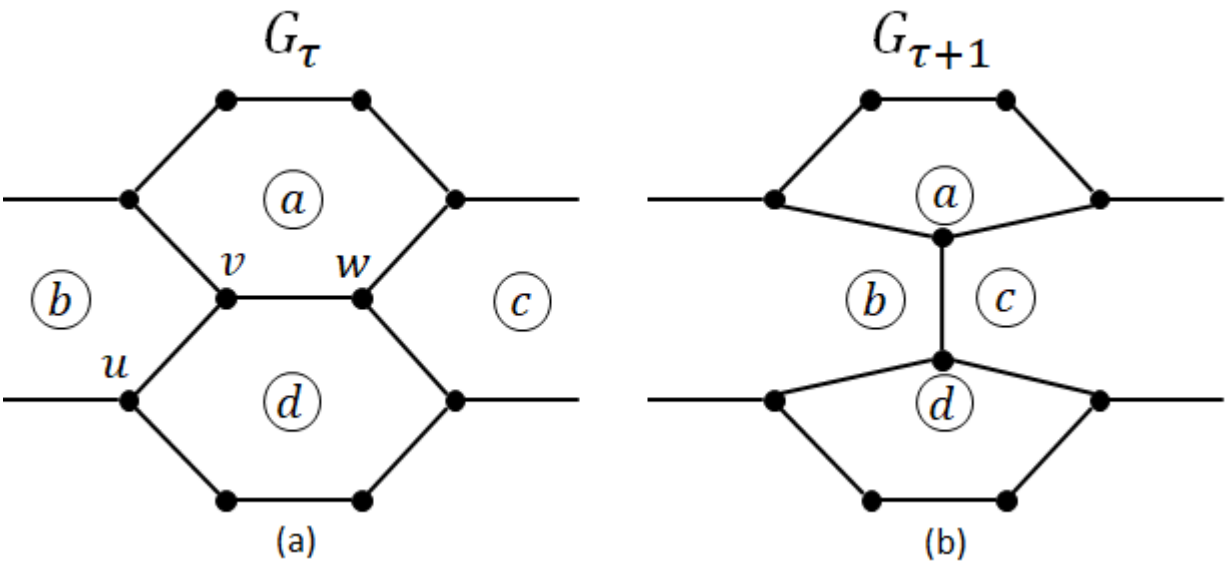


Figure 4.7: (a) A sample static graph, formed around mines a, b, c, and d. (b) Static graph (a) one time step later after mines b and c have moved towards each other. Vertices $v = (a, b, d)$ and $w = (a, c, d)$ 'disappear' after time step τ, while vertices (a, b, c) and (b, c, d) 'appear' at time step $\tau + 1$.

Function **NewEdge**(e, e', τ, M)\{
// Create TVG edge ε
$v \leftarrow$ **Tail**(e);
$v' \leftarrow$ **Head**(e');
$\vartheta \leftarrow (v, \tau)$;
$\vartheta' \leftarrow (v', \tau + 1)$;
$\mathcal{V} \leftarrow \mathcal{V} \cup \{\vartheta, \vartheta'\}$;
$\varepsilon \leftarrow (\vartheta, \vartheta')$;
if $v = s$ **or** $v' = t$ **then**
  $d_\varepsilon \leftarrow 0$;
  $\alpha_\varepsilon \leftarrow 0$;
  foreach $m \in M$ **do** $r_{\varepsilon m} \leftarrow 0$;
else
  // Calculate ship movement along ε and calculate derivatives
  $x_\varepsilon(\hat{\tau}) \leftarrow (\Delta x_{v'} - \Delta x_v)\,\hat{\tau}^2 + (v_x'^{\tau} + v_x^{\tau+1} - 2v_x^{\tau})\,\hat{\tau} + v_x^{\tau}$;
  $y_\varepsilon(\hat{\tau}) \leftarrow (\Delta y_{v'} - \Delta y_v)\,\hat{\tau}^2 + \left(v_y'^{\tau} + v_y^{\tau+1} - 2v_y^{\tau}\right)\hat{\tau} + v_y^{\tau}$;
  $\dfrac{d}{d\hat{\tau}} x_\varepsilon \leftarrow 2\,(\Delta x_{v'} - \Delta x_v)\,\hat{\tau} + v_x'^{\tau} + v_x^{\tau+1} - 2v_x^{\tau}$;
  $\dfrac{d}{d\hat{\tau}} y_\varepsilon \leftarrow 2\,(\Delta y_{v'} - \Delta y_v)\,\hat{\tau} + v_y'^{\tau} + v_y^{\tau+1} - 2v_y^{\tau}$;
  $\dfrac{d^2}{d\hat{\tau}^2} x_\varepsilon \leftarrow 2\Delta x_{v'} - 2\Delta x_v$;
  $\dfrac{d^2}{d\hat{\tau}^2} y_\varepsilon \leftarrow 2\Delta y_{v'} - 2\Delta y_v$;
  // Define length and acceleration along ε
  $d_\varepsilon \leftarrow \int_0^1 \sqrt{\left(\dfrac{d}{d\hat{\tau}} x_\varepsilon\right)^2 + \left(\dfrac{d}{d\hat{\tau}} y_\varepsilon\right)^2}\, d\hat{\tau}$;
  $\alpha_\varepsilon \leftarrow \sqrt{\left(\dfrac{d^2}{d\hat{\tau}^2} x_\varepsilon\right)^2 + \left(\dfrac{d^2}{d\hat{\tau}^2} y_\varepsilon\right)^2}$;
  // Calculate risk incurred from each mine m from traversing ε
  foreach $m \in M$ **do**
    $c_{\varepsilon m}(\hat{\tau}) \leftarrow \sqrt{(x_m(\hat{\tau}) - x_\varepsilon(\hat{\tau}))^2 + (y_m(\hat{\tau}) - y_\varepsilon(\hat{\tau}))^2}$;
    switch *on* $\hat{\tau}$ **do**
      case < 0
        $c_{\varepsilon m} \leftarrow c_{\varepsilon m}(\tau)$
      case > 1
        $c_{\varepsilon m} \leftarrow c_{\varepsilon m}(\tau + 1)$
      otherwise
        $c_{\varepsilon m} \leftarrow c_{\varepsilon m}(\hat{\tau})$;
      end
    end
    $p_{\varepsilon m} \leftarrow W(c_{\varepsilon m})$;
    $r_{\varepsilon m} \leftarrow -\log(1 - p_{\varepsilon m})$;
  end
end
\}

Algorithm 3: Pseudocode to generate new TVG edge ε and its associated parameters.

Unlike in the previous section, the edge $e = (u, v) \in E_\tau$ in Figure 4.7(a) cannot be used to generate an edge $\varepsilon \in \mathcal{E}$, as illustrated in Figure 4.6. This is because v and, thus, e, disappear before timestep $\tau + 1$. In order to prevent the vertex $(v, \tau) \in \mathcal{V}$ from becoming a dead end in $\mathcal{G}$, we find all vertices $\hat{v} \in V_{\tau+1}$ whose location at timestep $\tau + 1$ is within a distance γ of the location of v at timestep τ. The parameter γ is chosen to ensure a suitable $\hat{v}$ is found by defining it according to the farthest a vertex can move in a single time step:

$$\gamma \equiv \max_{m \in M, \tau = 1, \ldots, T} \sqrt{\left(m_x^\tau - m_x^{\tau+1}\right)^2 + \left(m_y^\tau - m_y^{\tau+1}\right)^2}. \tag{4.29}$$

For simplicity in Algorithm 4, we define

$$\mathbf{Dist}^\tau(v, \hat{v}) \equiv \sqrt{\left(v_x^\tau - \hat{v}_x^{\tau+1}\right)^2 + \left(v_y^\tau - \hat{v}_y^{\tau+1}\right)^2}. \tag{4.30}$$

For each $\hat{v}$ that satisfies $\mathbf{Dist}^\tau(v, \hat{v}) \leq \gamma$, if the edge $e' = (u, \hat{v})$ exists at timestep $\tau + 1$, i.e., $e' \in E_{\tau+1}$, then the edge $\mathbf{NewEdge}(e, e', \tau, M)$ is added to $\mathcal{E}$, as demonstrated in Figure 4.8(a). Figure 4.8(b) illustrates how the process is repeated if both endpoints disappear due to topological changes.

5.4 Journey Source and Sink

In the same fashion that **GenStatics** connects source and sinks verticies s and t to vertices $v \in V_t$ where a *path* can potentially begin or end, respectively; so must we connect source and sink verticies s_0 and t_{T+1} to verticies $\vartheta \in \mathcal{V}$ that represent where a potential *journey* can begin or end. Thus, we add edges to $\mathcal{E}$ of the form

$$\varepsilon = (s_0, \vartheta) \; \forall \vartheta = (s, \tau), \tau = 1 \ldots T, \tag{4.31}$$

$$\varepsilon = (\vartheta, t_{T+1}) \; \forall \vartheta = (t, \tau), \tau = 1 \ldots T. \tag{4.32}$$

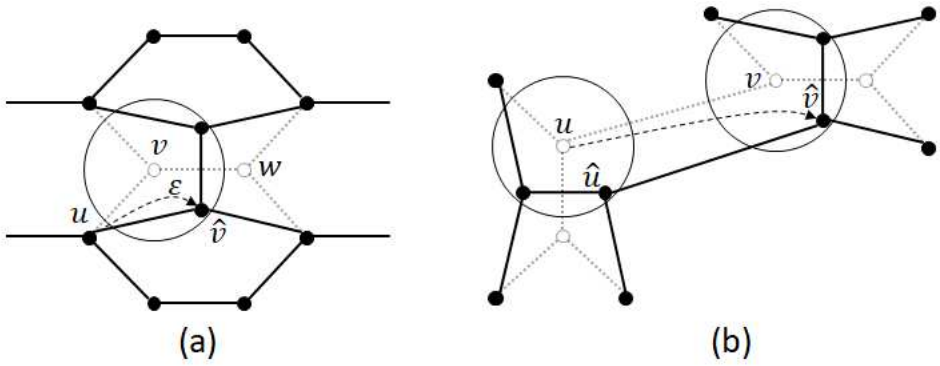


Figure 4.8: Two examples of topological events between timesteps τ and $\tau + 1$. Shaded edges are from E_τ and possess endpoints not in $V_\tau + 1$. To ensure connectivity of $\mathcal{G}$, these edges are "matched" to edges that appear in time step $\tau + 1$ by finding vertecies that appear within a distance γ (represented by a circle) of the disappearing vertex. In (a), edge $e = (u, v) \in E_\tau$ loses vertex v and is matched to edge $e' = (u, \hat{v}) \in E_{\tau+1}$ to generate $\varepsilon = \textbf{NewEdge}(e, e', \tau, M)$. In (b), both the tail and head of edge $e = (u, v) \in E_\tau$ disappear. Because vertecies $\hat{u}$ and $\hat{v}$ are both respectively within a distance γ of u and v at timestep τ, and $e' = (\hat{u}, \hat{v}) \in E_\tau$, the edge $\varepsilon = \textbf{NewEdge}(e, e', \tau, M)$ is added to $\mathcal{E}$.

As illustrated in Figure 4.9, these edges allow a journey $\mathcal{E}_{st}^{\tau\tau'}$ to begin and end at any time step such that $1 \leq \tau \leq \tau' \leq T$.

The resultant TVG $\mathcal{G}(\mathcal{V}, \mathcal{E})$, is the collection of edges between time steps, as exemplified in Figure 4.9(b). Combining the information contained in G_τ and M for $\tau = 1 \ldots T$ and the procedures defined in this section, the pseudocode for building the TVG $\mathcal{G}$ is presented in Algorithm 4.

5.5 Identify Minimum-Risk Journey

Once $\mathcal{G}$ has been generated, the final step is to identify the minimum-risk journey through $\mathcal{G}$. The function **GenTVG** ensures that $\mathcal{G}$ is a directed acyclic graph (DAG) with no dead-ends except s_0 and t_{T+1}. This structure permits the use of the Minimum Risk Path Algorithm (***MRPA***), presented by Richards et al. (2017), with several simple modifications.

```
Function GenTVG( ⋃_{τ=1...T} G_τ, M_τ ){
// Create TVG and journey source and sink
s_0 ← (s, 0);
t_{T+1} ← (t, T + 1);
V ← V ∪ {s_0, t_{T+1}};
E ← ∅;
for τ = 1 ... T − 1 do
    // Determine if topological changes occur
    foreach e = (u, u') ∈ E_τ do
        if u, u' ∈ P(V_τ) then
            // No change
            E ← E ∪ NewEdge(e, e, τ, M);
        else if u ∈ P(V_τ) then
            // Change eliminates u', search for new matching head
            foreach v' ∈ V_{τ+1} \ P(V_τ) s.t.
            Dist^τ(u', v') ≤ γ and e' = (u, v') ∈ E_{τ+1} do
                E ← E ∪ NewEdge(e, e', τ, M);
            end
        else if u' ∈ P(V_τ) then
            // Change eliminates u, search for new matching tail
            foreach v ∈ V_{τ+1} \ P(V_τ) s.t. Dist^τ(u, v) ≤ γ and e' = (v, u') ∈ E_{τ+1}
            do
                E ← E ∪ NewEdge(e, e', τ, M);
            end
        else
            // Event removes e, search for new matching edge
            foreach v ∈ V_{τ+1} s.t. Dist^τ(u, v) ≤ γ do
                foreach e' = (v, v') ∈ E_{τ+1} s.t.
                Dist^τ(u', v') ≤ γ and v, v' ∉ P(V_τ) do
                    E ← E ∪ NewEdge(e, e', τ, M);
                end
            end
        end
    end
    // Attach journey source and sink
    foreach ϑ ∈ V s.t. ϑ = (s, τ) do E ← E ∪ {(s_0, ϑ)} ;
    foreach ϑ ∈ V s.t. ϑ = (t, τ) do E ← E ∪ {(ϑ, t_{T+1})} ;
end
RETURN G(V, E);
}
```

Algorithm 4: Pseudocode to generate TVG $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Function **NewEdge** generates incident vertices.

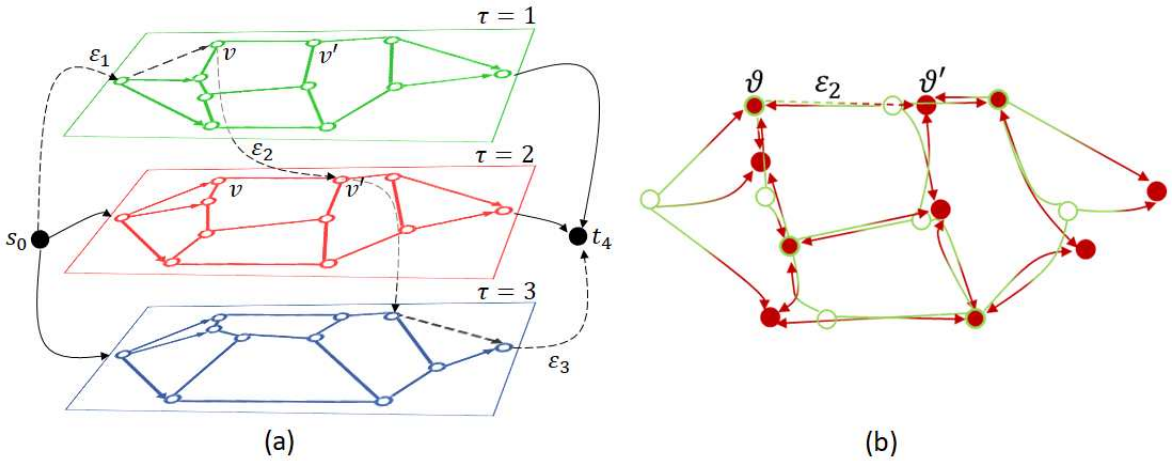


Figure 4.9: (a) A particular journey (dashed lines) through a TVG $\mathcal{G}(\mathcal{V}, \mathcal{E})$ constructed from static graphs G_1, G_2, and G_3. The edge $\varepsilon_2 = (\vartheta, \vartheta') \in \mathcal{E}$ such that $\vartheta = (v, 1)$ and $\vartheta' = (v', 2)$. (b) Edges $\varepsilon \in \mathcal{E}$ representing travel from $\tau = 1$ (green) to $\tau = 2$ (red) in TVG from (a).

First, ***MRPA*** uses a heuristic $\boldsymbol{H}$ to determine $\underline{r}_e$, a lower bound on the risk of traversing edge e, and uses a shortest-path algorithm to efficiently identify an initial path. Richards et al. (2017) defines the value of $\underline{r}_e$ based on n_m, the maximum number of edges a mine m can "threaten" in any given path, and M_e, the set of mines that threaten edge e . Because we define $r_{\varepsilon m}$ according to Washburn's function, which is asymptotic, n_m can grow to $|\mathcal{E}|$, and $M_\varepsilon = M \ \forall \varepsilon$. Thus, we redefine these parameters such that

$$M_\varepsilon \equiv m \in M : r_{\varepsilon m} \geq r_{min}, \tag{4.33}$$

where r_{min} is a measure of risk determined to be negligible. By setting Ω to the distance from at mine at which r_{min} is incurred, we define

$$n_m \equiv \left\lceil \frac{2\Omega}{d_{min}} \right\rceil, \tag{4.34}$$

where $d_{min} = \min_{\varepsilon \in \mathcal{E}} d_\varepsilon$. Using these parameters, we substitute $\underline{r}_e$ with

$$\underline{r}_{\varepsilon} \equiv \sum_{m \in M_{\varepsilon}} \frac{r_{\varepsilon m}}{n_m}, \tag{4.35}$$

into $\boldsymbol{H}$ in order to identify an initial journey $\widehat{\mathcal{E}}_{st}^{\tau\tau'}$.

Next, ***MRPA*** reduces the size of $\mathcal{G}$ by removing vertices $\vartheta \in \mathcal{V}$ and edges $\varepsilon \in \mathcal{E}$ that cannot be part of a minimum-risk journey. We apply the preprocessing steps $\boldsymbol{P}$ and $\boldsymbol{R}$, as defined by Richards et al. (2017), with the following adjustments: In addition to removing "high-risk" edges, we modify $\boldsymbol{P}$ to also remove edges $\varepsilon \in \mathcal{E}$ where

$$\alpha_{\varepsilon} \geq \alpha_{max}. \tag{4.36}$$

The parameter α_{max} is defined as the maximum acceleration a ship navigating $\mathcal{G}$ can achieve, making the associated edges unnavigable. The heuristic $\boldsymbol{P}$ is also amended to remove vertices from $\mathcal{V}$ which are not reachable by source s_0 or that cannot reach sink t_{T+1}. Using a simple non-weighted version of Dijkstra's algorithm, for each vertex $\vartheta = (v, \tau) \in \mathcal{V}$, the minimum number of edges from s_0 and to t_{T+1} are set to ϑ_s and ϑ_t, respectively. The amended $\boldsymbol{P}$ then checks

$$\vartheta_s + \vartheta_t \leq T. \tag{4.37}$$

If Equation (4.37) does not hold, a journey cannot include vertex ϑ in the timeframe allotted and is removed.

With regards to $\boldsymbol{R}$, because its ability to reduce $|\mathcal{E}|$ depends upon the existence of "risk-free" vertices, we adjust $\boldsymbol{R}$ for our application so as to define an edge $\varepsilon \in \mathcal{E}$ to be "risk-free" if

$$r_{\varepsilon m} \leq r_{min} \; \forall m \in M. \tag{4.38}$$

Lastly, ***MRPA*** executes an A* search **ExtendPath** to recursively evaluate journeys in $\mathcal{G}$ and identify a minimum-risk journey. Starting at $s_0 = (s, 0)$, **ExtendPath** creates a

sub-journey $\mathcal{E}_{s\vartheta}^{\tau\tau'}$ one edge at a time, using the threat-additive risk $\mathcal{R}_{\mathcal{E}_{s\vartheta}^{\tau\tau'}}$ incurred by the sub-journey and bounds determined by $\boldsymbol{H}$ to determine if extending the sub-journey along a particular edge $\varepsilon = (\vartheta, \vartheta')$ could possibly lead to a journey that ends at $t_{T+1} = (t, T+1)$ with less risk than the current incumbent solution $\widehat{\mathcal{E}}_{st}^{\tau\tau'}$. In addition to the penalty **ExtendPath** uses to avoid unnecessary maneuvers and encourage a smaller $\mathcal{D}_{\mathcal{E}_{st}^{\tau\tau'}}$, we apply a small penalty to the number of time steps required to complete a journey, as measured by $|\mathcal{E}_{st}^{\tau\tau'}|$, in order to avoid journeys that take an unnecessarily long amount of time. We refer to our journey-modified version of ***MRPA*** as "***jMRPA***", and provide summary psuedocode in Algorithm 5.

Function ***jMRPA***$\left(M, \bigcup_{\tau=0\ldots T} P^{\tau}, \bigcup_{b=1\ldots B} \rho_b, \overrightarrow{\delta}_b, \mathcal{S}, \mathcal{T}, r_{min}, n_m, \alpha_{max}\right)\{$
// Predict future locations of mines
GenFields$\left(M, \bigcup_{\tau=0\ldots T} P^{\tau}\right)$;
// Generate "snapshot" graphs G_τ
GenStatics$\left(M, \bigcup_{b=1\ldots B} \rho_b, \overrightarrow{\delta}_b, \mathcal{S}, \mathcal{T}\right)$;
// Generate TVG $\mathcal{G}$
GenTVG$\left(\bigcup_{\tau=1\ldots T} G_\tau, M\right)$;
// Find initial journey $\widehat{\mathcal{E}}_{st}^{\tau\tau'}$
$\boldsymbol{H}(\mathcal{G}, r_{min}, n_m)$;
// Reduce $\mathcal{V}$ and $\mathcal{E}$
$\boldsymbol{P}(\mathcal{G}, \alpha_{max}, T)$;
$\boldsymbol{R}(\mathcal{G}, r_{min})$;
// Find the minimum-risk journey in $\mathcal{G}$
ExtendPath$(\mathcal{G}, s_0, \emptyset)$;
$\}$

Algorithm 5: Psuedocode for ***jMRPA***. Algorithm returns the minimum-risk journey $\mathcal{E}_{st}^{\tau\tau'}$ through the TVG $\mathcal{G}$ along with its associated risk, length, and duration.

6 Computational Results and Analysis

To code ***jMRPA***, we use Matlab 8.3.0.532 (R2014a) to generate $\mathcal{G}(\mathcal{V}, \mathcal{E})$, while graph reductions and journey finding is done in C++ using the Library for Efficient Modeling and Optimization in Networks (LEMON) from the open source Computational Infrastructure for Operations Research (COIN-OR) (Dezső et al. 2011). Code is executed on an Intel i5 1.60GHz with 8 GB of RAM. To test ***jMRPA***, we consider the scenario in which entrance to MacDill AFB is obstructed via free-floating minefields (FFM) in Tampa Bay (Figure 4.10). Current and wind conditions are taken from National Oceanic and Atmospheric Administration (NOAA) (National Oceanic and Atmospheric Administration 2017) buoy observations from 07-04-2017, and minefield density, defined by $|M|$, is varied such that $|M| = \{10, 20, 30, 40, 50, 60, 70\}$. Not only do minefields of this size and density accurately reflect the range of historical real-world encounters (Finlan 2004), but testing shows that minefields above this range represent instances in which risk is above acceptable levels, i.e., 0.1 or more. We generate five instances for each minefield density and, for each instance, randomly determine mine locations within the AO by choosing $x-$ and $y-$coordinates from a uniform distribution. We use six-minute timesteps based on weather station reporting intervals. We vary T between 5 and 40, and begin our time frames at midnight.

6.1 Solution Quality

Table 4.1 summarizes the average characteristics, across all test instances instances of a particular $|M|$ and T (see Section 5.2 for choice of T), for the generated TVG $\mathcal{G}$ and the optimal journey found by ***jMRPA***.

Statistics for the 'Original' TVG reflect $\mathcal{G}$ as generated by **GenTVG**, while 'Reduced' reflects the application of preprocessing steps, $\boldsymbol{H}$, $\boldsymbol{P}$, and $\boldsymbol{R}$. With respect to the journey found by ***jMRPA***, Velocity indicates the final velocity of a ship beginning its journey at rest. Average risk remains below 0.10 for the journeys we find. Because we measure risk based on the probability of *any* detonation, regardless of ship proximity, these values represent reasonable and acceptable mission parameters. In comparison to the 8 nautical

Figure 4.10: Tampa Bay leading to MacDill AFB. Initial mine locations for a particular instance are indicated by dots. The initial static graph G_1 approximates the AO. Source and sink s and t model attempted entry to MacDill AFB to the Northeast.

miles (nm) required to sail directly through the AO depicted in Figure 4.10, optimal journeys require less than 2 additional nautical miles on average to maneuver and avoid mines. Furthermore, all velocities are well within the capabilities of a large naval vessel navigating a minefield, e.g., the MCM-1 Avenger Mine Countermeasure Ship (Sherman 1999).

Table 4.1: Average TVG and optimal journey statistics across all 5 instances for each $|M|$ and its associated number of time steps allowed T. Columns 5 and 6 reflect characteristics of $\mathcal{G}$ once the original TVG has be reduced via $\boldsymbol{H}$,$\boldsymbol{P}$, and $\boldsymbol{R}$. A vaulue of zero in column 5 indicates that the initial solution found by $\boldsymbol{H}$ possesses zero risk and allows $\boldsymbol{R}$ to remove all edges. Column 9 is final velocity of a ship starting at rest.

Instance		$\mathcal{G}(\mathcal{V},\mathcal{E})$				$\mathcal{E}_{st}^{\tau\tau'}$		
$\|M\|$	T	Original		Reduced		$\mathcal{R}_{\mathcal{E}_{st}^{\tau\tau'}}$	$\mathcal{D}_{\mathcal{E}_{st}^{\tau\tau'}}$ (nm)	Velocity (kts)
		$\|\mathcal{E}\|$	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{V}\|$			
10	10	1260	560	0	2	0	9.7	5.4
20	10	1800	761	0	2	0	8.3	2.6
30	20	5223	2021	435	190	0.01	9.4	5.4
40	20	6259	2389	193	104	0.01	9.9	5.8
50	30	12710	4513	1932	511	0.06	11.2	10
60	30	12746	4666	1352	505	0.07	9.9	7.4
70	40	19371	7037	1323	417	0.04	10.4	10.9

6.2 Run Times

The average time ***jMRPA*** spends identifying an optimal journey through $\mathcal{G}$, reported in Table 4.2, remains relatively low across all $|M|$. The time required to generate $\mathcal{G}$, however, is much larger. Table 4.2 shows that TVG generation can take upwards of one hour.

The time required to generate $\mathcal{G}$ is primarily dictated by $|\mathcal{E}|$, the number of TVG edges. For each edge, determining its characteristics, i.e., parametric equations, risk, length and acceleration, requires solving the many derivatives and integrals defined in Algorithm 3. The number of edges, in turn, is dictated by $|M|$, T, and the number of topological changes that occur. The number of mines $|M|$ is a physical trait of the FFM and cannot be altered. Furthermore, given the set of static graphs G_τ, the number of topological changes is dictated by the choice of T. The number of allowable time steps T can be varied, but only to a certain point. Table 4.3 demonstrates the effect of varying T for a particular minefield with $|M| = 50$.

Table 4.2: Average time across all instances taken by **GenTVG** to generate $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and identify the minimum-risk journey $\mathcal{G}(\mathcal{V}, \mathcal{E})$ for the given values of $|M|$ and T.

$\lvert M\rvert$	T	$\mathcal{G}(\mathcal{V}, \mathcal{E})$ Generation (min)	$\mathcal{E}_{st}^{\tau\tau'}$ Identification (min)
10	10	3	<0.1
20	10	5	<0.1
30	20	13	<0.1
40	20	18	<0.1
50	30	38	1.5
60	30	53	2.1
70	40	80	0.3

As Table 4.3 demonstrates, reducing T can greatly reduce $|\mathcal{E}|$ and the time required to generate $\mathcal{G}$; however, choosing a T that is too small can also eliminate potential lower-risk journeys. Because the topology of the static Voronoi graphs G_τ dictate a minimum number of edges required to connect the source and sink, for the FFM measured in Table 4.3, a journey is not feasible for $T \leq 15$. Each subsequent increase to T introduces additional, lower-risk journeys. Values of T in Table 4.1 were chosen to be the minimum value capable of obtaining journeys with acceptable risk, i.e, less than 0.10.

Table 4.3: Effects of varying T for a particular minefield with $|M| = 50$. Colums 2 and 3 reflect the total edges and subsequent generation time. Column 4 reports the risk associated with the optimal journey found through the resultant TVG. Values of "-" represent no path found.

T	$\lvert\mathcal{E}\rvert$	Time (min)	$\mathcal{R}_{\mathcal{E}_{st}^{\tau\tau'}}$
10	3,456	15	-
15	5,365	18	-
20	7,281	22	0.23
25	9,112	28	0.16
30	10,985	38	0.06

7 Conclusions And Future Work

We address the growing concern posed by free-floating minefields by developing a time-varying graph (TVG) through which time-dependent journeys can be found. Minimum-risk journeys are identified using a threat-additive model, specially modified to account for the parametrically defined paths of both ship and mines. While previous researchers have discussed the theoretical bounds of a TVG composed of moving-point Voronoi graphs, we present a unique and practical approach to construction. The continuous paths Voronoi vertices follow cannot be easily expressed functionally. The heuristics we develop that ensure TVG connectivity, along with the continuous parametric equations that define navigable journeys, represent one of the first successful discretizations to efficiently generate networks that provide real-world, low-risk journeys. In all instances, modified preprocessing functions are able to reduce the size of the TVG so that an A* search algorithm can identify the minimum risk journey in an average of two minutes or less. The largest challenge faced by our procedure is the time required for TVG generation. While this can be somewhat mitigated by intelligently choosing T, several additional factors may offer significant improvement with minimal impact to solution quality.

Figure 4.11 provides the physical path of an optimal journey overlaid on the initial static graph G_τ. As time moves forward, the optimal journey increasingly deviates from paths possible in the static graph. However, by determining the time step τ' before which serious drift occurs, it may be possible to determine the first portion of a journey, up to this time step, using only information from G_1 with negligible effect to risk. Subsequently, ***jMRPA*** would only need to generate $\mathcal{G}$ for time steps $\tau \geq \tau'$.

Second, in order to allow partial circumnavigation of mines, at each time step the static graph G_τ includes two oppositely-directed edges to represent each Voronoi edge. In the solutions we find, however, journeys rarely include substantial retrograde movement as this tends to invite more risk. By eliminating static edges which move away from the sink t by some minimum distance, up to half of the edges in the TVG may be removed.

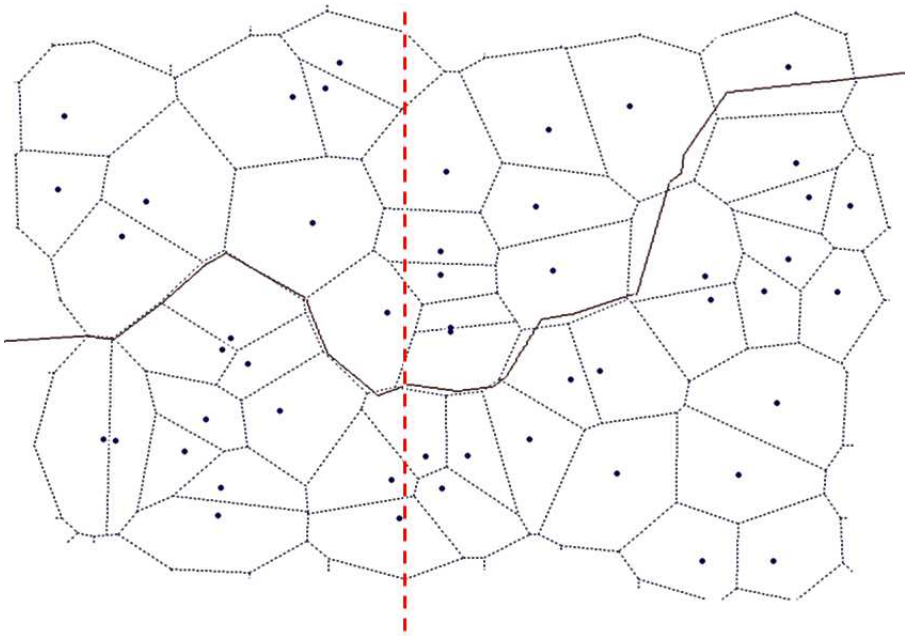


Figure 4.11: Physical path of an optimal journey (solid line) overlaid on initial static graph G_τ (dotted lines). In this instance, significant path variation does not occur until approximately the eighth time step (vertical dashed line).

Lastly, a large portion of the edges in each TVG are added to maintain connectivity in response to topological events between time steps. These events occur when a static edge is small enough that it may disappear due to mine movement. In many of these cases, the length of the edge in question is so small that it could be replaced with a single vertex connected to its incident edges. While this would result in the static graphs G_τ to become degenerate, i.e., possess vertices with degree higher than three, accounting for this complication would greatly reduce the number of edges induced by topological changes. Furthermore, fewer superfluous short edges would also allow ***jMRPA*** to more quickly complete journeys and, thus, more quickly prune non-optimal sub-journeys from its search.

Although we use environmental, i.e., wind and current, data to predict mine movement and generate our TVG, our methodology is applicable to any threat-avoidance situation in which threat movement can be modeled. Of particular interest to the author is flight

planning of unmanned USAF assets. Judd and McLain (2001) presents a model for finding minimum-risk navigable path through a series of *stationary* threats. However, mobile Surface-to-Air Missile (SAM) platforms like the Russian 2K12 Kub (Cordesman Anthony and Wagner 1990) are capable of quick relocation and deployment, and, as such, are increasingly employed by countries such as Iran to effectively deny large airspaces (Mashregh News 2011) in a fashion similar to how drifting minefields deny access to ports. Using the operational range of mobile SAMS and an enemy movement prediction model such as the one developed by Ownby and Kott (2006) should allow our procedure to generate minimum-risk flight journeys for UAV's.

8 Acknowledgments

We would like to thank Dr. Alexandra Newman for her assistance in writing this paper, and John Ford for his insights into navigating Voronoi diagrams.

CHAPTER 5
CONCLUSION

Since the submission of "Optimal Course Scheduling for United States Air Force Academy Cadets," the Pathways to Excellence team has continued to move forward with the model we have provided. In addition to greatly reducing the costs involved in the acquisition process in which they are currently engaged, faculty have expressed increasing excitement at the ability to move away from the archaic "M-T" alternating-day schedule of calls. With increased scrutiny on the necessity of military academies (Fleming 2015), faculty are looking forward to keep pace with other Tier 1 schools without sacrificing their military identity.

This past year, the results from "Minimum-Risk Routing Through a Mapped Minefield" not only took first place at the Colorado School of Mines MEGN 503 presentation competition, but was lauded at the INFORMS 2016 Annual Meeting by members of the Turkish Navy. Researchers at the Middle East Technical University are currently using our research in their work on "Daily Aircraft Routing For Amphibious Ready Groups."

The code we develop in "Navigating Free-Floating Minefields via Time-Evolving Voronoi Graphs" is noteworthy as it is one of the first times that practicalities of connecting and employing a time-varying Voronoi graph have been defined. Until now, the majority of papers concerning moving Voronoi graphs have focused either on their theoretical characteristics, or on practical utilization of the plane partition they provide. Many prior researchers in the field of robotics have used static Voronoi graphs to implement code that finds real-world preferred *routes*; our research takes some of the first practical steps to extend this work to allow *journeys* to be found and executed.

REFERENCES

Air Force A3/5, 2017. "Plans and Requirements Directorate CAMIS Modernization Estimates." Interview.

Albers, G., Guibas, L. J., Mitchell, J. S. and Roos, T., 1998. "Voronoi diagrams of moving points." *International Journal of Computational Geometry & Applications*, **8**(03): 365–379.

Alsmadi, O. M. K., Za'er, S., Abu-Al-Nadi, D. I. and Algsoon, A., 2011. "A novel genetic algorithm technique for solving university course timetabling problems." In *Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on*, IEEE, 195–198.

American Bureau of Shipping, 2006. *Guide for Vessel Maneuverability*. American Bureau of Shipping.

Aubin, J. and Ferland, J. A., 1989. "A large scale timetabling problem." *Computers & Operations Research*, **16**(1): 67–77.

Aycan, E. and Ayav, T., 2009. "Solving the course scheduling problem using simulated annealing." In *Advance Computing Conference, 2009. IACC 2009. IEEE International*, IEEE, 462–466.

Babaei, H., Karimpour, J. and Hadidi, A., 2015. "A survey of approaches for university course timetabling problem." *Computers & Industrial Engineering*, **86**: 43–59.

Babel, L. and Zimmermann, T., 2015. "Planning safe navigation routes through mined waters." *European Journal of Operational Research*, **241**(1): 99–108.

Bakir, A. and Aksop, C., 2008. "A 0-1 integer programming approach to a university timetabling problem." *Hacettepe Journal of Mathematics and Statistics*, **37**(1): 41–55.

Bakolas, E. and Tsiotras, P., 2010. "Optimal pursuit of moving targets using dynamic Voronoi diagrams." In *Decision and Control (CDC), 2010 49th IEEE Conference on*, IEEE, 7431–7436.

Bekker, J. and Schmid, J., 2006. "Planning the safe transit of a ship through a mapped minefield." *ORION*, **22**(1): 1–18.

Bernitt, T. R. and Tangredi, S. J., 2002. "Mine warfare and globalization: Low-tech warfare in a high-tech world." *Globalization and Maritime Power*.

Boerman, D., 1994. *Finding an Optimal Path Through a Mapped Minefield*. Master's thesis, Department of Operations Research, Naval Postgraduate School, Monterey, California.

Boland, N., Hughes, B., Merlot, L. and Stuckey, P., 2008. "New integer linear programming approaches for course timetabling." *Computers and Operations Research*, **35**(7): 2209–2233.

Branum, D., 2014. "Pathways to excellence team envisions changes to improve cadets' schedules." *United States Air Force Academy News*.

Breivik, Ø. and Allen, A. A., 2008. "An operational search and rescue model for the Norwegian Sea and the North Sea." *Journal of Marine Systems*, **69**(1): 99–113.

Breivik, Ø., Allen, A. A., Maisondieu, C. and Roth, J. C., 2011. "Wind-induced drift of objects at sea: The leeway field method." *Applied Ocean Research*, **33**(2): 100–109.

Brown, G., Dell, R. and Wood, K., 1997. "Optimization and persistence." *Interfaces*, **5**(27): 15–37.

Burke, E., Elliman, D., Ford, P. and Weare, R., 2005. "Examination timetabling in British universities - a survey." *Lecture Notes in Computer Science*, **1153**: 76–90.

Byers, T. and Waterman, 1984. "Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming." *Operations Research*, **32**(6): 1381–1384.

Caprara, A., Fischetti, M. and Toth, P., 1999. "A heuristic method for the set covering problem." *Operations Research*, **47**(5): 730–743.

Carlyle, M., Royset, J. and Wood, R., 2009. "Routing military aircraft with a constrained shortest-path algorithm." *Military Operations Research*, **14**(3): 31–52.

Carlyle, W., Royset, J. and Wood, R., 2008. "Lagrangian relaxation and enumeration for solving constrained shortest-path problems." *Networks*, **52**(4): 256–270.

Casteigts, A., Flocchini, P., Quattrociocchi, W. and Santoro, N., 2012. "Time-varying graphs and dynamic networks." *International Journal of Parallel, Emergent and Distributed Systems*, **27**(5): 387–408.

Changsheng, L., 2005. "The hidden dragon in the deep: The present situation and development of PLA mine weaponry." *World Outlook*, (9): 32.

Cheng, E., Kruk, S. and Lipman, M., 2003. "Flow formulations for the student scheduling problem." *Lecture Notes in Computer Science*, **2740**(1): 299–309.

CNO, 1991. *The United States Navy in Desert Shield/Desert Storm*. Naval Historical Center, Washington DC, Appendix A: Chronology ed., `http://www.history.navy.mil/wars/dstorm/index.html`.

Cordesman Anthony, H. and Wagner, A. R., 1990. "The lessons of modern war: Volume II-the Iran-Iraq war." *Boulder, CO: Westview*.

Cormen, T. H., 2009. *Introduction to Algorithms*. MIT press.

Cornish, G., 2003. *U.S. Naval Mine Warfare Strategy: Analysis of the Way Ahead*. U.S. Army War College, Carlisle, PA.

Dempsey, M., 2014. "Joint concept for entry operations." `http://www.dtic.mil/doctrine/concepts/joint_concepts/jceo.pdf`.

Department of the Navy, 1996. *Operationial Characteristics of U.S. Naval Mines (U)*. ORD 696(B), Washington 25, D.D.

Devillers, O., Golin, M., Kedem, K. and Schirra, S., 1996. "Queries on Voronoi diagrams of moving points." *Computational Geometry*, **6**(5): 315–327.

Dezső, B., Jüttner, A. and Kovács, P., 2011. "LEMON–an open source C++ graph template library." *Electronic Notes in Theoretical Computer Science*, **264**(5): 23–45.

Dinstein, Y., 2016. *The conduct of hostilities under the law of international armed conflict*. Cambridge University Press.

Dostert, M., Politz, A. and Schmitz, H., 2016. "A complexity analysis and an algorithmic approach to student sectioning in existing timetables." *Journal of Scheduling*, **19**(3): 285–293.

Duncan, R. C., 1962. "America's use of sea mines." Tech. rep., DTIC Document.

Eagle, J., 2008. *Lateral Range Curves/Sweepwidth*. Naval Postgraduate School, Monterey, CA, 4th ed., Lecture Notes.

Easton, K., Nemhauser, G. and Trick, M., 2001. "The travelling tournament problem: Description and benchmarks." *Lecture Notes in Computer Science*, **2239**: 580–585.

Edelsbrunner, H., 2014. "Voronoi and Delaunay diagrams." In *A Short Course in Computational Geometry and Topology*, Springer, 9–15.

Edelsbrunner, H., Tan, T. S. and Waupotitsch, R., 1992. "An $\mathcal{O}(n^2 \log n)$ time algorithm for the minmax angle triangulation." *SIAM Journal on Scientific and Statistical Computing*, **13**(4): 994–1008.

Eglese, R. and Rand, G., 1987. "Conference seminar timetabling." *Journal of the Operational Research Society*, **38**(7): 591–598.

Erickson, A. S., Goldstein, L. J. and Murray, W. S., 2009. "Chinese mine warfare: A PLA navy assassin's mace capability (China Maritime Study, number 3)." Tech. rep., DTIC Document.

Feng, X., Lee, Y. and Moon, I., 2016. "An integer program and a hybrid genetic algorithm for the university timetabling problem." *Optimization Methods and Software*, 1–25.

Ferreira, A., 2004. "Building a reference combinatorial model for MANETs." *IEEE Network*, **18**(5): 24–29.

Finlan, A., 2004. *The Royal Navy in the Falklands Conflict and the Gulf War: Culture and Strategy*. Frank Cass Publishers, Portland, OR.

Fleming, B., 2015. "Let's abolish West Point: Military academies serve no one, squander millions of tax dollars." `http://www.salon.com/2015/01/05/lets_abolish_west_point_military_academies_serve_no_one_squander_millions_of_tax_dollars/`.

Fletcher, B. and Wernli, R. L., 2003. "Expanding missions for small unmanned undersea vehicles (UUVs)." In *ASME 2003 22nd International Conference on Offshore Mechanics and Arctic Engineering*, American Society of Mechanical Engineers, 599–602.

Fong, C. W., Asmuni, H., McCollum, B., McMullan, P. and Omatu, S., 2014. "A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems." *Information Sciences*, **283**: 1–21.

Fortune, S., 1987. "A sweepline algorithm for Voronoi diagrams." *Algorithmica*, **2**(1-4): 153–174.

Fourer, R., Gay, D. and Kernighan, B., 2003. *AMPL - A Modeling Language For Mathematical Programming*. Thomson Brooks/Cole, second ed.

Freedberg, S., 2012. "Navy mine war game 'not about Iran,', officially; 'biggest mine exercise ever'." *Breaking Defense*.

Gonzalez, G., 2011. *Optimizing Semester Assignment of United States Air Force Acedmy Cadets*. Ph.D. thesis, Colorado School of Mines, Golden, Colorado.

Har-Peled, S. and Raichel, B., 2014. "On the complexity of randomly weighted Voronoi diagrams." In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, ACM, 232.

Hart, P. E., Nilsson, N. J. and Raphael, B., 1968. "A formal basis for the heuristic determination of minimum cost paths." *IEEE transactions on Systems Science and Cybernetics*, **4**(2): 100–107.

Hertz, A., 1991. "Tabu search for large scale timetabling problems." *European Journal of Operational Research*, **54**(1): 39–47.

HQ USAFA, 2012. "Cadet administrative management information system assignment and scheduling algorithms: Final report." Tech. Rep. Unclassified, United States Air Force Academy.

IBM, 2010. *IBM ILOG AMPL Version 12.1 Users guide: Standard (Command-line) Version Including CPLEX Directives*. Incline Village, NV.

Inanc, T., Muezzinoglu, M., Misovec, K. and Murray, R., 2008. "Framework for low-observable trajectory generation in presence of multiple radars." *Journal of Guidance, Control, and Dynamics*, **31**(6): 1740–1749.

Jaumard, B., Semet, F. and Vovor, T., 1998. "A generalized linear programming model for nurse scheduling." *European Journal of Operational Research*, **107**: 1–18.

Judd, K. B. and McLain, T. W., 2001. "Spline based path planning for unmanned air vehicles." In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, vol. 9.

Karelahti, J., Virtanen, K. and Ostrom, J., 2008. "Automated generation of realistic near-optimal aircraft trajectories." *Journal of Guidance, Control, and Dynamics*, **31**(3): 674–688.

Khonggamnerd, P. and Innet, S., 2009. "On improvement of effectiveness in automatic university timetabling arrangement with applied genetic algorithm." In *Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. Fourth International*

Conference on, IEEE, 1266–1270.

Kim, J. and Hespanha, J., 2003. "Discrete approximations to continuous shortest path: Application to minimum-risk path planning for groups of UAVs." In *42nd IEEE Conference on Decision and Control*, Maui, Hawaii, 1734–1740.

Klotz, E. and Newman, A., 2013. "Practical guidelines for solving difficult mixed integer linear programs." *Surveys in Operations Research and Management Science*, **18**(1): 18–32.

Kongsberg Maritime AS, 2011. "The Complete Modular and Portable MCM System." `https://www.km.kongsberg.com/ks/web/nokbg0397.nsf/AllWeb/78B0EA89879D4FC5C125785B00475A63/$file/Kongsberg_HUGIN_MCM_folder_v1.pdf`.

LaGrone, S., 2014. "A terrible thing that waits (under the ocean)." `http://www.popsci.com/blog-network/shipshape/terrible-thing-waits-under-ocean`.

LaRivee, D., 2017. "Future Schedule of Calls Response." USAFA Memorandum.

Lewis, R., 2008. "A survey of metaheuristic-based techniques for university timetabling problems." *OR Spectrum*, **30**(1): 167–190.

Li, P.-C., 2009. *Planning the Optimal Transit For A Ship Through A Mapped Minefield*. Master's thesis, Department of Operations Research, Naval Postgraduate School, Monterey, California.

Lin, P., Bekey, G. and Abney, K., 2008. "Autonomous military robotics: Risk, ethics, and design." Tech. rep., DTIC Document.

Lübbecke, M. E., 2015. "Comments on: An overview of curriculum-based course timetabling." *TOP*, **23**(2): 359–361.

Luedtke, J., Ahmed, S. and Nemhauser, G. L., 2010. "An integer programming approach for linear programs with probabilistic constraints." *Mathematical programming*, **122**(2): 247–272.

Maldonado, L. A., 1996. *A methodology for evaluating mine actuation data*. Ph.D. thesis, Monterey, California. Naval Postgraduate School.

Mashregh News, 2011. "Looking to enhance the battle capabilities of the corps." `http://www.mashreghnews.ir/news/56646/`.

McNally, B., 2012. "The future of mine countermeasures." [Online; posted 21-May-2012].

Mercer, G. N. and Sidhu, H. S., 2007. "Two continuous methods for determining a minimal risk path through a minefield." In *Proceedings of the 13th Biennial Computational Techniques and Applications Conference*, W. Read and A. J. Roberts, eds., James Cook University Townsville, North Queensland, Australia, vol. 48, C293–C306.

Miller, A. J. and Wolsey, L. A., 2003. "Tight formulations for some simple mixed integer programs and convex objective integer programs." *Mathematical Programming*, **98**(1): 73–88.

MirHassani, S. and Habibi, F., 2013. "Solution approaches to the course timetabling problem." *Artificial Intelligence Review*, 1–17.

Morton, D. P., Pan, F. and Saeger, K. J., 2007. "Models for nuclear smuggling interdiction." *IIE Transactions*, **39**(1): 3–14.

Muhandiramge, R., Boland, N. and Wang, S., 2009. "Convergent network approximations for the continuous Euclidean length constrained minimum cost path problem." *SIAM Journal on Optimization*, **20**(1): 55–77.

Muljowidodo, K., Sapto Adi, N., Budiyono, A. and Prayogo, N., 2009. "Design of SHRIMP ROV for surveillance and mine sweeper." *Indian Journal of Marine Sciences*, **38**(3): 332–337.

Müller, T. and Murray, K., 2010. "Comprehensive approach to student sectioning." *Annals of Operations Research*, **181**(1): 249–269.

Nandy, S., 2009. "Voronoi diagrams." Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata. Lecture.

National Oceanic and Atmospheric Administration, 2017. "Tides and currents products." `https://tidesandcurrents.noaa.gov/products.html`.

Naval Studies Board, 2001. *Naval mine warfare: Operational and technical challenges for naval forces*. National Academies Press, Washington, D.C.

NCES, 2015. "Nces 2015 011." *Digest of Education Statistics*, 105.50.

Nehme, M., 2009. *Two-Person Games for Stochastic Network Interdiction: Models, Methods, and Complexities*. Ph.D. thesis, University of Texas at Austin, Austin, Texas.

Ocean Studies Board, 2000. *Oceanography and mine warfare*. National Academies Press, Washington, D.C.

Osborn, K., 2017. "Navy Knifefish undersea drone finds mines in test." `https://defensesystems.com/articles/2017/02/03/knifefish.aspx`.

Ownby, M. and Kott, A., 2006. "Reading the mind of the enemy: Predictive analysis and command effectiveness." Tech. rep., Solers, Inc., Arlington, VA.

Peng, H. and Yucai, L., 2002. "Outline of undersea weaponry."

Phillips, A. E., Waterer, H., Ehrgott, M. and Ryan, D. M., 2015. "Integer programming methods for large-scale practical classroom assignment problems." *Computers & Operations Research*, **53**: 42–53.

Plumer, B., 2017. "America's staggering defense budget, in charts." *Washington Post*, **1**.

Razoux, P., 2015. *The Iran-Iraq War*. Harvard University Press.

Richards, C., Odom, C., Morton, D. and Newman, A., 2017. "Minimum-risk routing through a mapped minefield.", manuscript submitted for publication.

RWM Italia S.p.A., 2012. *MANTA antilanding shallow water mine*. Rheinmetall Defence, Ghedi, Italy.

Sampson, S. E., Freeland, J. R. and Weiss, E. N., 1995. "Class scheduling to maximize participant satisfaction." *Interfaces*, **25**(3): 30–41.

Sampson, S. E. and Weiss, E. N., 1995. "Increasing service levels in conference and educational scheduling: A heuristic approach." *Management Science*, **41**(11): 1816–1825.

Schaerf, A., 1999. "A survey of automated timetabling." *Artificial Intelligence Review*, (13): 87–127.

Sherman, R., 1999. "MCM-1 Avenger mine countermeasure ships." `https://fas.org/man/dod-101/sys/ship/mcm-1.htm`.

Shields, R., 2012. "Cultural topology: The seven bridges of Königsburg, 1736." *Theory, Culture & Society*, **29**(4-5): 43–57.

Shome, S., Nandy, S., Pal, D., Das, S., Vadali, S., Basu, J. and Ghosh, S., 2012. "Development of modular shallow water AUV: Issues & trial results." *Journal of The Institution*

of Engineers (India): Series C, **93**(3): 217–228.

Shorack, G., 1964. "Algorithms and analog computers for the most reliable route through a network." *Operations Research*, **12**(4): 632–633.

Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N. and Feyzeau, P., 2013. "Path planning: A 2013 survey." In *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, IEEE, 1–8.

Stewart, J., 2015. *Single variable calculus: Early transcendentals*. Nelson Education.

Sun, W. and Tsiotras, P., 2017. "Sequential pursuit of multiple targets under external disturbances via Zermelo–Voronoi diagrams." *Automatica*, **81**: 253–260.

Szturomski, B., 2015. "The effect of an underwater explosion on a ship." *Zeszyty Naukowe Akademii Marynarki Wojennej*, **56**(201): 57–73.

Tang, J., Musolesi, M., Mascolo, C. and Latora, V., 2010. "Characterising temporal distance and reachability in mobile and online social networks." *ACM SIGCOMM Computer Communication Review*, **40**(1): 118–124.

Technical Manual, 1996. *Explosive Safety Precautions for Research Vessels*. NAVORD Op 3696.

Tedeschi, D., 2015. "Is it time to abolish the U.S. Air Force?" *Air & Space Magazine*.

Truver, S., 2011. "Taking mines seriously." In *Mine Warfare in China's Near Seas*, Naval Institute Press.

Tuga, M., Berretta, R. and Mendes, A., 2007. "A hybrid simulated annealing with Kempe chain neighborhood for the university timetabling problem." In *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, IEEE, 400–405.

Turner, M., 2016. "National commission on the future of the army." In *Hearings before the House Armed Services Committee Subcommittee on Tactical Air and Land Forces*, 114th U.S. Congress.

Tutte, W. T., 1958. "A homotopy theorem for matroids, I." *Transactions of the American Mathematical Society*, **88**(1): 144–160.

U.S. Congress, 1992. "Conduct of the Persian Gulf War: Final report to Congress." In *102nd Congress, April*.

U.S. Joint Chiefs of Staff, 2006. *Joint Operations Planning*. Joint Publication 5-0. Washington DC: U.S. Joint Chiefs of Staff.

U.S. Navy, 1959. *Operational Characteristics of U.S. Naval Mines (UNCLASSIFIED)*. Department of Defense.

Washburn, A., 2007. *Mine Warfare Models*. Naval Postgraduate School, Monterey, CA.

Wren, A., 1996. *Practice and Theory of Automated Timetabling: Scheduling, Timetabling and Rostering - A Special Relationship?*, vol. 1153. Springer.

Zabarankin, M., Uryasev, S. and Pardalos, P., 2002. "Optimal risk path algorithms." In *Cooperative Control and Optimization*, P. Pardalos, D. Hearn and R. Murphey, eds., Springer US, vol. 66 of *Applied Optimization*, 273–298.

Zwick, U., 2001. "Exact and approximate distances in graphs a survey." In *European Symposium on Algorithms*, Springer, 33–48.