MODELING AND OPTIMIZATION OF CONTAINER INSPECTION SYSTEMS

by

CHRISTINA M. YOUNG

A Dissertation submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Industrial and Systems Engineering

written under the direction of

Professor Elsayed A. Elsayed

and approved by

________________________

________________________

________________________

________________________

New Brunswick, New Jersey

October, 2010
Container inspection is vital to maintaining secure ports-of-entry and preventing undesired cargo from entering the United States. The inspection process can be generalized as the collection and analysis of information obtained from multiple sensors. Formulating a mathematical model of container inspection allows for evaluation and improvement of the process. The performance of the system under a specified policy is evaluated using one or more objectives such as misclassification errors (false accept and false reject), costs associated with these errors, inspection cost, inspection time, and others. The main contributions of this research are the modeling, formulation, and optimization of inspection policies under different conditions. Furthermore, the dissertation introduces a new class of problems in scheduling theory in which the allocation of inspections is not defined and appears as a decision variable in the solution.

In the initial model, the overall system decision is a Boolean function of the individual station decisions. Under these conditions we define an optimal sequence of stations with respect to the expected cost of inspection and solve simultaneously for the threshold level
values and sequence of stations that produce a minimum total cost. This optimization is extended to include the time for inspection as an objective and a multi-objective optimization approach is developed. Next we introduce an independent error term that accounts for measurement error contributed by the sensor and propose some strategies, including repeat inspection, to improve the system’s performance. We investigate an approach to approximating the efficient frontier for three objectives.

We then consider distinct risk categories and due times for containers. Approaches are developed to determine the optimal allocation and scheduling of inspection operations to minimize false acceptance and tardiness objectives. The problem is presented as a variation of the open shop scheduling problem with no predefined operations. A solution approach to this simultaneous allocation and scheduling problem is proposed and its performance is compared with an enumerative approach. The results show that the proposed approach produces near-optimal solutions in a much shorter time than full enumeration and is capable of solving large problems for which the enumerative approach is intractable.
Acknowledgements

I would like to acknowledge and thank my advisor, Dr. Elsayed A. Elsayed for his advice, encouragement, patience, and confidence in me over these many years. There is no way I could have done this without him.

My wonderful husband Chris has provided infinite amounts of love, support, and understanding; I simply can’t thank him enough.

I would also like to thank my committee members Dr. Susan L. Albin, Dr. David W. Coit, and Dr. Fred S. Roberts for their time and valuable suggestions.

I am grateful to my family and friends who may have never quite understood what I was doing, but supported me wholeheartedly nonetheless. Thank you, I love you all.

I have cherished my time at Rutgers and I’d like to thank the many people throughout the years who have made it worth remembering.

Finally I appreciate the financial support from FAA through a graduate fellowship and ONR, NSF, and NSA for funding parts of the research in this dissertation.
Table of Contents

ABSTRACT OF THE DISSERTATION ................................................................. II

ACKNOWLEDGEMENTS ............................................................................. IV

TABLE OF CONTENTS ................................................................................. V

LISTS OF TABLES ........................................................................................ VIII

LIST OF FIGURES ......................................................................................... IX

CHAPTER 1 INTRODUCTION ....................................................................... 1

1.1 OVERVIEW OF PORT-OF-ENTRY INSPECTION .................................... 1
1.2 BACKGROUND OF POE INSPECTION .................................................. 3
1.3 ORGANIZATION OF THE DISSERTATION .......................................... 6

CHAPTER 2 LITERATURE REVIEW ............................................................... 10

2.1 PORT-OF-ENTRY RESEARCH .............................................................. 10
2.2 THRESHOLD AND SEQUENCE OPTIMIZATION .................................. 13
2.3 MINIMIZATION OF TOTAL EXPECTED COST ...................................... 17
2.4 TREATMENT OF MEASUREMENT ERROR .......................................... 19
2.5 SUMMARY ........................................................................................... 21

CHAPTER 3 SEQUENCE AND THRESHOLD LEVEL OPTIMIZATION .......... 24

3.1 PROBLEM DESCRIPTION ..................................................................... 25
3.1.1 Port-of-Entry Container Inspection System .................................... 25
3.1.2 Sensor Performance and Inspection Threshold Levels ...................... 26
3.1.3 System Inspection Policy ............................................................... 29
3.2 OPTIMIZATION APPROACHES ......................................................... 30
3.2.1 Minimization of Cost of Misclassification ....................................... 30
3.2.2 Optimal Sequence for Expected Inspection Cost ............................ 38
3.2.3 Minimization of Expected Total Cost ............................................ 42
6.4.4 Complete Enumeration................................................................. 112
6.5 NUMERICAL EXAMPLES ..................................................................... 113
   6.5.1 Computation Time ........................................................................ 114
   6.5.2 Solution Performance Comparison ................................................ 122
6.6 DISCUSSION ...................................................................................... 127

CHAPTER 7 CONCLUSIONS ..................................................................... 129

REFERENCES ......................................................................................... 133

APPENDIX A .......................................................................................... 140

APPENDIX B .......................................................................................... 144

APPENDIX C .......................................................................................... 146

APPENDIX D .......................................................................................... 147

APPENDIX E .......................................................................................... 154

APPENDIX F .......................................................................................... 161

APPENDIX G .......................................................................................... 165

CURRICULUM VITAE ............................................................................... 175
Lists of Tables

Table 3.1  Distribution Parameter Values for Sensors in Parallel-Series Example ........ 49
Table 4.1.  Examples of Pareto Optimal Solutions.......................................................... 66
Table 4.2.  Optimal Solution of Design 19 ................................................................. 70
Table 4.3.  Analysis of Variance Results......................................................................... 72
Table 5.1.  Inspection Policy Performance Comparison.................................................... 87
Table 6.1.  Computation Time of CE for Special Case with $k = 3$, $dc = 4$ ................. 114
Table 6.2.  Computation Time of CE for Special Case with $k = 4$, $dc = 4$ ................. 115
Table 6.3.  Computation Time of CE for General Case with $dc = 7$ ............................. 116
Table 6.4.  Computation Time of CE for General Case under Various $wT$ Values ....... 117
Table 6.5.  Computation Time of GH for Special Case with $dc = 4$ ............................. 118
Table 6.6.  Computation Time of GH for General Case with $dc = 7$ ............................. 119
Table 6.7.  Computation Time of GH for General Case under Various $wT$ Values ...... 119
Table 6.8.  Computation Time of GH for General Case with $k = 3$, $dc = n$ ............... 120
Table 6.9.  Computation Time of GH for Special Case Large Problems ....................... 121
Table 6.10. GH Solution Performance for Special Case with $k = 3$ ............................ 123
Table 6.11. GH Solution Performance for Special Case with $k = 4$ ............................ 123
Table 6.12. GH Solution Performance for Special Case under Mixed Due Times ...... 124
Table 6.13. GH Solution Performance for Special Case under Various $w_T$ Values ...... 124
Table 6.14. GH Solution Performance for General Case with $k = 3$ .......................... 125
Table 6.15. GH Solution Performance for General Case with $k = 4$ .......................... 125
Table 6.16. GH Solution Performance for General Case under Mixed Due Times ...... 126
Table 6.17. GH Solution Performance for General Case under Various $w_T$ Values ..... 126
List of Figures

Figure 3.1. Conceptual Depiction of Parallel-Series System.......................... 32
Figure 3.2. Conceptual Depiction of Series-Parallel System.......................... 33
Figure 3.3. ROC Curve of a Parallel System of Three Sensors......................... 37
Figure 3.4. Minimum Cost Curves for Parallel System.................................. 46
Figure 3.5. Minimum Cost Curves for Series System.................................... 47
Figure 3.6. Minimum Cost Curves for Parallel-Series System.......................... 48
Figure 3.7. Minimum Cost Curves for Series-Parallel System.......................... 49
Figure 3.8. Budget Curve for Parallel System........................................... 50
Figure 4.1. Comparison of Solution Methods to Multi-Objective Problem........... 65
Figure 4.2. GA Method Results.................................................................... 67
Figure 4.3. Boxplots of Four Factors for Equally Weighted Cost and Time........... 71
Figure 5.1. Performance of Simple Inspection with Varying Measurement Error..... 85
Figure 5.2. Performance of Repeat Inspection with Varying Re-inspection Band..... 86
Figure 6.1. Lower Bound Algorithm............................................................ 105
Figure 6.2. Graph of Computation Times of GH for Special Case Large Problems.... 121
CHAPTER 1

INTRODUCTION

1.1 Overview of Port-of-Entry Inspection

Each year shipping containers make more than 200 million trips across international sea lanes, and more than 95 percent of the non-North American foreign trade arrives into US ports by ship. Increase in trade agreements, growth in the world economy, and the outsourcing of manufactured goods have significantly increased maritime traffic. The value of export goods produced and transported globally in 2000 was about $6.186 trillion (World Trade Organization 2001). Global commerce is heavily dependent on the movement of shipping containers, which carry a wide range of materials, food, equipment and other types of products and commodities. Disruption of maritime shipping due to security issues would have a profound impact on the world economy. This is magnified by the increasing dependence of companies on just-in-time delivery of raw material and components and extensive supply chain networks between suppliers and companies.

The United Nations passed several resolutions with the objective of improving security in maritime trade. Likewise, the United States initiated the Container Security Initiative (CSI) to ensure container security through different approaches starting from the origin
port of the container and ending at the delivery port in the United States. The inspection of containers may occur at the port of origin, port of entry, and/or enroute. The selection of containers for inspection may be random, based on a “risk” factor, or a mix of the two.

Emphasis on improving the security of containers prompted the development and installation of a wide range of inspection machines and sensors. These machines or sensors have different capabilities of detecting the presence of nuclear materials, biological agents, drugs, hazardous materials, and other illegal shipments. New security improvements at ports-of-entry require expensive new or retrofitted infrastructure and technology and re-trained personnel. New security measures also have major impacts on the cost of container shipping and handling. Both for the container terminal operators and the vessel operators it is paramount to minimize “turn-around time”, i.e. the loading and discharging of containers should be done as quickly as possible (Henesey et al. 2002). An average container liner spends 60% of its time in port and has a cost of $1000 per hour or more (Rebollo et al. 2000). Recently, Wein et al. (2006) provide a detailed report of the cost components associated with port security, operation and the effect of delay and state that the hourly waiting cost of a containership arriving at its U.S. port of debarkation is tens of thousands of dollars. To shorten the time spent by vessels, terminal operators need to emphasize resource allocation and reduce the security related inspection processes. Improving security through container inspection has a profound effect on the operations of the shipping supply chain. For example, additional inspection time at a port is likely to lead to a delivery delay (Wein et al. 2006), which in turn tends to increase safety inventories and inventory costs and hence reduce supply chain
productivity. Clearly there are two conflicting objectives: improved security and improved productivity; the challenge is to balance both. The type of inspections, number of containers to be inspected, and the inspection policy have a profound effect on the cost of the system, risk of accepting undesired containers and potential delays and congestion at the ports.

1.2 Background of POE Inspection

The container inspection process can be generalized as the collection and analysis of information obtained from multiple sensors and other sources to decide whether to allow a container to pass through the port. Screening is the first step and is generally used to select some containers for further inspection. For instance, the Automated Targeting System (ATS) associates data in the cargo manifest with intelligence and other potential indicators of a container’s heightened risk (Wein et al. 2007).

A typical inspection system begins with radiation detection. Containers are driven through a Radiation Portal Monitor (RPM) at approximately five miles per hour, where radiation emissions are detected. The equipment is passive, so it absorbs radiation from containers as they pass through the RPM. A graphic profile of the radiation reading is produced and analysis is performed to determine if the profile suggests the presence of radioactive material. If a container’s profile is deemed suspicious, the container may be subjected to further inspection to determine the source of radiation. This is usually accomplished by using a lightweight hand-held Radiation Isotope Identification Device
(RIID) or an Advanced Spectroscopic Portal (ASP) to identify the radiation isotope. The RIID is more sensitive than the traditional Geiger counter, and takes an isotope reading to determine the type of radiation being emitted. The RIID is capable of distinguishing between naturally occurring, harmless radiation (emitted by materials such as: ceramic tile, granite, kitty litter, fertilizer, or food products containing potassium, including bananas or avocados) and weapons grade radioactive and nuclear materials, including highly enriched uranium or plutonium used in nuclear devices (USCBP 2006).

Image scanning is a widely used technique of non-invasively “seeing” into a container (James et al. 2002). This technique is utilized in the Vehicle and Cargo Inspection System (VACIS), which is a gamma-ray imaging system that produces a two-dimensional image of container contents. This allows operators to identify contraband, verify shipping manifests, and recognize potentially dangerous items. The system can scan a container in less than a minute, and human analysis of each scan requires two minutes (Wein et al. 2007). The image is produced by a radioactive source and detector tower that travel along fixed parallel tracks on either side of the container to be inspected. There is also a mobile truck-mounted version which allows for flexible deployment.

In addition to inspecting containers for radioactive material and image analysis, the inspection stations may identify biological warfare agents by using biosensors, currently under research and ready for deployment on an experimental basis, that can detect trace amounts of viral or bacterial pathogens in situ and provide immediate analysis (Graham and Sabelnikov 2004). Other equipment for detecting biological agents uses fluorescent
particle counters for detecting airborne bacteria. In this case the threshold level of the
decision is related to the count of these particles. Likewise, methods for detection of
chemical agents, currently sarin, cyanide, and pesticides, may be applied at other
inspection stations using different sensors (Murrar and Southard 2002). Finally Wein et
al. (2006) states that several technologies can be used to detect a nuclear weapon and a
variety of newer technologies are undergoing rapid development.

A new technology called nuclear resonance fluorescence imaging (NRFI) is capable of
non-intrusively measuring the isotopic content of the material in a space by exposing it to
a continuous energy distribution of photons and detecting the scattering to that of known
isotopes and dangerous materials (Bertozzi and Ledoux 2005). In another port-of-entry
station an X-ray system generates a high energy X-ray beam that traverses the container
under inspection. A detector collects X-ray energy from the container and provides a
signal which is processed to detect the presence of very high X-ray attenuation within the
container, indicative of nuclear weapons or materials (Annis 2002).

Following inspection a container is classified as acceptable (containing no suspicious
material) or not. The consequence of such decision generates two types of errors, Type I
and II, which are defined by the following hypotheses. The null hypothesis $H_0$ is that a
given container is safe, and the alternate hypothesis $H_1$ is that it contains suspicious
material. Type I error (also known as an "error of the first kind" or $\alpha$ error) is referred to
as a "false reject" and corresponds to instances where a container is rejected or goes
through extensive manual examination when in fact it has no suspicious contents. Type
II error (also known as an "error of the second kind" or $\beta$ error) is referred to as a "false accept" and corresponds to instances where a container that has suspicious contents is accepted.

1.3 Organization of the Dissertation

In this dissertation we consider a port-of-entry (POE) container inspection system where containers are inspected through a sequence of different stations to detect the presence of radioactive materials, biological and chemical agents, and other illegal cargo. The stations are modeled as independent sensors; each returns a measurement that corresponds to one specific attribute. The limit applied to a measurement (raw number or obtained through signal processing) to make an accept/reject decision about a particular container attribute is represented as a threshold level. The threshold level is a set value in the domain of sensor measurements and has a direct impact on how a container is classified. The inspection policy, which includes the sequence in which sensors are applied and the threshold levels used at the inspection stations, affects the probability of misclassifying a container as well as the cost and time spent in inspection.

Clearly, the accuracy of inspection in terms of passing (or accepting) those containers which do not have illegal cargo and rejecting those which indeed contain such cargo with minimum delays depends on many factors such as the acceptance threshold levels of the sensors, sequence in which the stations are implemented, precision of the instruments, and others. In the case where containers have stated due times, the allocation and
scheduling of inspection operations affect system tardiness, another important objective. Therefore, this dissertation investigates different formulations of mathematical models of container inspection in order to evaluate and improve the inspection process. The formulations consider different conditions and inspection policies. This work represents a unique approach to the formulation of the port-of-entry inspection problem as an analytical model.

The dissertation is organized as follows. Chapter 1 presents an overview of the POE inspection and its importance. It describes the inspection technologies used in detection of key attributes, such as the presence of radioactive material in the container, and defines the dissertation objectives. Literature relevant to the POE inspection problem and formulations, analysis, and methodologies for investigating this problem is surveyed in Chapter 2.

Chapter 3 starts by modeling an inspection system with a Boolean decision function combining station results. The threshold level values and sequence of inspection are optimized to achieve the minimum expected total cost of misclassification errors and inspection. We also find that the optimal sequence with respect to expected cost of inspection can be defined and used in the overall optimization. The research in this section is mostly based on our published work in Elsayed et al. (2009).

Next we consider the time required for inspection in Chapter 4. The problem becomes a multi-objective problem with two objectives: minimization of the total inspection cost
which includes the cost of inspection and cost of misclassifying containers, and the minimization of the total expected time a container spends in the inspection system. A multi-objective optimization approach is developed to determine the optimal sensor arrangement and threshold levels for these objectives. The research in this section is mostly based on our published work in Young et al. (2010).

Chapter 5 introduces a measurement error term that is independent of the natural variation in container attribute readings. Measurement error has a negative impact on the classification capabilities; this effect is investigated with an updated model. Repeat inspection is also considered, as it effectively reduces the impact of measurement error because the same attribute value is repeated and the effect of the random measurement error term averages to zero. We analyze and attempt to express the effect of repeat inspection on an inspection policy.

In Chapter 6 we consider the problem where containers have specified due times and associated tardiness penalties. The timeframe in which containers are available for inspection according to their due times is taken into account by placing the total system tardiness in the objective function. We also assume that the containers have associated risk profiles that describe the risk of unacceptable cargo classified into risk categories. A new formulation is introduced that considers multiple, independent risk categories and the feasible scheduling of inspection operations. We develop methods to determine the optimal inspection allocation and schedule that minimizes container tardiness and the probability of false acceptance. Finally in Chapter 7 we summarize the research and
propose directions for future work.
CHAPTER 2

LITERATURE REVIEW

In this chapter we present a review of the research on the POE inspection problem and its associated subject areas. We begin with a section on the current research and issues related to the investigation and modeling of the POE problem followed by a section that addresses the challenging problems of container inspection sequence and the threshold levels for sensors at inspection stations. Reviews of cost minimization of container inspection and inclusion of the sensors’ measurement errors in the models are discussed in details in the following sections.

2.1 Port-of-Entry Research

Recently there has been significant interest in the modeling and investigation of POE systems, including inspection of containers. Although there many aspects and issues related to POE systems we limit our review to research on the POE container inspection problem, which considers the risk of potential smugglers attempting to bring illegal cargo into the country. Brandenstein (1995) provides the highlights of the role risk assessment played in the United States technology program for nonintrusive inspection of cargo containers for illicit drugs. Koch (2007) creates a port simulation model to investigate the effect of the introduction of new inspection technologies on the overall port
operations, while Lee et al. (2008) and Lewis et al. (2003) develop complex models that optimize port throughput and illustrate the effect of container inspection procedures on port operations. Lewis et al. (2003) develop a best-first heuristic search procedure to model the problem of moving containers from inbound ships to staging areas where security inspections can occur and moving containers from staging areas and areas where security inspections have been completed to outbound ships but do not consider the performance of the inspection operation itself. The treatment of optimal resource allocation in reliability systems in Azaiez and Bier (2007) parallels the optimum sequence of attribute inspection in container inspection. Dye (2003) summarizes some basic requirements of inspection systems: 1. Sensor systems must be operationally practical and must provide information that can enable effective, preemptive actions to be taken, 2. Sensors systems must be highly sensitive, providing a low probability of missed detections (false negatives) and 3. Sensor systems must give a low probability of false alarms (false positives). Many researchers focus on choosing best way to select containers and assign random inspection (allocation of inspection) since slowing the flow of containers through a port long enough to inspect either all or a statistically significant random selection of imports would be economically intolerable (Loy and Ross 2002). Researchers have investigated the problem of container inspection with different objectives.

In the investigation of the POE inspection problem, the inspection system is considered as a collection of stations (these could be sensor equipment or other instruments) where each is dedicated to the detection of a specific characteristic of undesirable material in the
container. The inspection procedure consists of independent sequential inspections at stations, the results of which determine whether to pass a container or subject it to “manual” inspection. The container may leave the sequence of inspection stations when some conditions are met or it may continue through other stations until completion of the entire inspection system. Stroud and Saeger (2003) address the problem as a binary decision tree problem. Madigan et al. (2007) extend the work of Stroud and Saeger by incorporating the threshold levels of the inspection sensors and develop a novel binary decision tree search algorithm that operates on the space of potentially acceptable trees. They describe computationally more efficient approaches for this binary decision tree problem and obtain optimum sensor threshold levels that minimize the total cost of the inspection system. Boros et al. (2009) have formulated the problem as a large linear programming problem with continuous readings. Ramirez-Marquez (2008) also investigates the process as a binary decision tree and uses a probabilistic solution discovery algorithm to generate an optimal strategy. Elsayed et al. (2009) consider the problem of selecting threshold levels and the sequence of inspections to minimize the expected total, and Young et al. (2010) extend the problem to minimize the expected cost and time. Wein et al. (2006) model the inspection problem using a game-theoretic approach where the players are the border protection agency (whose objective is to maximize probability of detection) and the smugglers of undesired material (whose objective is to minimize the probability of detection). Wein et al. (2006) develop a mathematical model to find the optimal inspection strategy subject to port constraints and a budget. Kantor and Boros (2007) also use a game theoretic approach but apply randomization strategies, also known as mixed strategies, to improve the detection rate.
Wein et al. (2007) model the specific problem of deploying a particular type of station, RPM, at overseas ports and formulate a queueing model to determine the optimum number of monitors that maximizes the probability of detection subject to resource constraints.

The problem of container inspection at POE can also be viewed as a screening process with similarities to passenger and baggage screening at airports. Both are concerned with potential misclassification, with false acceptance more costly than false rejection. This problem is investigated by Canadalino et al. (2004), where they introduce a comprehensive cost function that includes direct costs associated with the purchase and operation of baggage screening security devices and the indirect costs associated with device errors. They present a methodology to determine the best selection of baggage screening security devices that minimizes the expected annual total cost of a baggage screening strategy (Canadalino et al. 2004). Jacobson et al. (2001) defines an NP-complete decision problem to determine a system response function (which translates device responses to alarm or clear signal) that minimizes the false alarm rate while meeting a stated false clear rate.

### 2.2 Threshold and Sequence Optimization

The function of the POE inspection system is to detect illegal cargo based on the measurements or analysis obtained from sensors; therefore a decision rule is defined to reject or accept containers according to their measurements. In developing the model we
assume that historical distributions of sensor measurements from acceptable and unacceptable containers are known. Under the decision rule two errors are possible: reject an acceptable container or accept an unacceptable one. A Neyman-Pearson criterion can be used to model systems with these two types of errors (Scott 2004) and derive an optimal decision rule, for instance, to minimize the probability of false rejection given an upper bound on the probability of false acceptance (Jacobson et al. 2001). Thomopoulos et al. (1989) consider a Neyman-Pearson formulation where they assume a bound on the global probability of false alarm and the goal is to determine the optimum local and global decision rules that minimize the global probability of false clear. When inspection stations are deployed so that their observations are independent, it can be shown that the decision rules are threshold rules based on likelihood ratios (Thomopoulos et al. 1989). The problem now becomes one of determining the optimal thresholds of the sensors at each station. While this task is quite non-trivial, it can still be done for a reasonably small number of inspection stations using iterative techniques or using complete enumeration (Stroud and Saeger 2003, Ding et al. 2006, Elsayed and Zhang 2006, Zhang et al. 2006). The related concept of determining the optimum threshold levels that minimizes the total expected cost including making the “wrong” decision is addressed by the maintenance problem in Liao et al. (2006).

There is also the question of the sequence in which to implement the inspections to minimize expected cost. If the threshold levels of sensors at inspection stations are fixed, determining the optimal sequence of container inspection is known as the sequential diagnosis problem. In sequential diagnosis, the components of a system are inspected
one by one to find out the functionality of the system (Unluyurt 2004). This is analogous to the attributes of a container being inspected in a fixed sequence in order to determine the status of the container and return a classification. An inspection strategy $S$ is a rule that specifies on the basis of the states of the attributes already inspected which characteristics are to be inspected next or stops inspection by recognizing the correct status of the container. Such an inspection strategy can naturally be represented as a binary decision tree and is well documented in literature (Unluyurt 2004). In the sequential diagnosis problem the dependency of the system’s state on the states of its components is assumed to be known, either defined by the system function $f(x)$ or via an oracle. In the POE inspection problem this is equivalent to defining the acceptability of a container based on the true contents, that is, if the exact cargo of a container is truly known there would be no ambiguity as to whether it is permitted entry through the port because the verdict is assumed to be clearly defined by applicable laws and customs restrictions.

The problem of determining the optimum inspection sequence has been investigated by numerous researchers. Many variations of the general sequential diagnosis problem are classified as NP-complete (Hyafil and Rivest 1976). Branch and bound and dynamic programming formulations have been proposed to solve the general problem of sequential diagnosis, both of which run in exponential time in general. A dynamic programming approach has been proposed in (Cox et al. 1989) for threshold functions. Other approaches such as genetic algorithms and artificial intelligence have been utilized. Since the general problem is NP-hard in many cases, researchers have focused their
investigation on known system configurations such as series, parallel, series-parallel, parallel-series and \( k \)-out-of-\( n \) (presence of \( k \) attributes out of total \( n \)) systems in order to obtain optimum solutions for a small number of attributes.

In certain classes of Boolean system functions, determination of the optimum sequence of inspection is formulated and investigated using approaches parallel to those used in the optimal sequential inspection procedure for reliability systems -also called least-expected-cost failure-state diagnosis- as described in (Butterworth 1972, Halpern 1974a, 1974b, 1977, Ben-Dov 1981, Cox et al. 1989, 1996). Zhang et al. (2006) and Azaiez and Bier (2007) define the optimal sequence for series, parallel, and combined series/parallel systems (including series-parallel and parallel-series). Bioch and Ibaraki (1995) and Chang et al. (1990) propose polynomial time algorithms that produce optimal solutions for a \( k \)-out-of-\( n \) system. These algorithms are generalized further in Boros and Unluyurt (1999) by providing a general algorithm that is optimal for double regular systems (having identical components).

In general the objective of the sequential diagnosis problem is to find a strategy for a given system that minimizes the expected cost of testing, which equates to one of the objectives in the POE inspection problem. However in the POE problem we assume imperfect inspections which result in certain probabilities of misclassifying containers—that is, rejecting a truly acceptable container or accepting a truly unacceptable one. These two types of errors both have associated costs, which are significant compared to the cost of performing the inspections. The next section discusses research that minimizes the
total expectation of these three costs in an inspection process.

2.3 Minimization of Total Expected Cost

Raouf et al. (1983) develop a model for multi-characteristic inspection to minimize the total expected cost from inspection, false rejection, and false acceptance. Lee (1988) simplifies the model and extended it to cases with random probabilities of defectives. In both papers it is assumed that the inspections are imperfect and the probabilities of type I and type II errors are known, the incoming quality of characteristics is known, and the costs of false acceptance, false rejection, and inspection at each stage are known.

Studies such as these involving the inspection of multi-characteristic critical components are closely related to the POE inspection problem. We assume that a container has multiple attributes which indicate different types of prohibited cargo and the presence of any prohibited cargo at all makes the entire container unacceptable. This is parallel to a component being “classified as nondefective only if all the characteristics meet the quality specifications” (Lee 1988). Duffuaa and Al-Najjar (1995) define a component as critical if “it causes a disaster or a very high cost upon failure,” which is assumed to occur when a defective unit is accepted and then fails in operation. The parallel of this definition in the POE inspection context would be allowing illegal or dangerous cargo to pass through the port undetected, the consequences of which would fit the above description. Many studies note that in the case of critical components, the cost of false
acceptance is much higher than the cost of false rejection and this is also true for POE inspection.

Similar to Raouf et al. (1983) and Lee (1988), Duffuaa and Raouf (1990) find the optimal sequencing of stages and the number of cycles to minimize the expected total cost, where the inspection of all characteristics constitutes a cycle, the inspection of each characteristic is referred to as a “stage” in the cycle, and more than one cycle may be required for optimum performance. The optimal sequence of inspections found in these studies corresponds to the optimal defined for sequential diagnosis in a series system. Since the failure of any characteristic results in overall failure, the component behaves as a series system.

Duffuaa and Al-Najjar (1995) present a new plan of repeat cycles within each inspection stage - so that repeated inspections of the same characteristic are performed before proceeding to the next characteristic - and find the optimal number of repeat cycles and sequence of inspections to minimize the total expected cost. In a subsequent paper Duffuaa and Al-Najjar (1997) relax an assumption and allow for a different number of repeat inspections for different characteristics. However, whether the cycles of repeated inspections are carried out within stages (Duffuaa and Al-Najjar 1995, 1997) or encompassing stages (Raouf et al. 1983, Lee 1988, Duffuaa and Raouf 1990), the repeated inspections are modeled as independent and rely on probabilities of misclassification which are assumed to be known. In our approach to the POE problem we use a more complex model with historical distributions of attribute measurement
values and an independent error term which makes repeated inspections of the same attribute dependent on the initial readings. Research related to measurement error in repeated inspections is discussed in Section 2.4.

All of the above mentioned papers in this section develop a model of the inspection process in which they assume the inspection error (which we refer to as misclassification errors) is known. Clearly the optimization of any inspection plan is dependent on the values given for inspection error. Duffuaa and Khan (2005) illustrate the impact of varying inspection error rates on the performance measures in optimized inspection plans. We define and solve a broader problem in which inspection errors are functions of the decision variable of sensor threshold level, which is optimized simultaneously with the inspection sequence.

2.4 Treatment of Measurement Error

Random measurement error inherent in the process of obtaining sensor readings may result in significant misclassification of containers, both false acceptance and false rejection. Mader et al. (1999) evaluate the economic impacts of measurement error inspection plans. Chandra and Schall (1988) model inherent measurement error as a random variable independent from the true value of interest and analyze the effect of repeated readings of the same measurement. They use an estimator of the true value based on Schweppe (1973) which combines the values of repeated measurements.
Chandra and Schall (1988) find that the probabilities of making a wrong decision can be minimized by taking more than one reading of the measurement of interest.

There are many sources that contribute to measurement errors including human errors, gauge errors, and environments. It is important to identify sources and reduce their contributions but in many cases it is difficult, if not impossible, to do so. Chandra and Schall (1988) note that measurement error cannot be completely avoided. Kim et al. (2007) categorize research efforts to reduce the impact of measurement error into two approaches. The first approach is improvement in the precision of measurement devices. Chen and Chung (1996) and Tang and Schneider (1988) address the optimal selection of measurement precision level in designing economical inspection strategies. The second approach is based on the use of guard bands to identify “good” and “bad” items. The economic impact of guard bands is investigated by Eagle (1954), Grubbs and Coon (1954), and Hutchinson (1991). Deaver (1995) provides a comparative study of several strategies for the use of guard bands. McCarville and Montgomery (1996) use an experimental design approach for finding the optimal guard bands for serial gauges. Recently, Kim et al. (2007) integrate these two approaches and simultaneously determine both the optimum precision level and guard band in order to reduce the impact of measurement errors and minimize the total expected cost.
2.5 Summary

A thorough review of the literature indicates that the POE inspection problem provides challenging research areas to be investigated. Container inspection bears many similarities to the inspection of items for conformance to specification. Johnson et al. (1991) note that inspection is “an essential component of quality control methodology” while Al-Sultan and Rahim (1997) state that inspection may be used in removing defective items in a population and is an important part of a quality control program. The area of quality control has concepts that are relevant to the container inspection problem. Items are declared conforming or non-conforming, parallel to containers being pronounced acceptable or unacceptable for entrance into a port according to a threshold level. The inspection process is subject to error with known probabilities of detection in both problems. They also share a common root in hypothesis testing; research on inspection in quality control considers two kinds of errors, false-positive and false-negative, equivalent to the type I and type II errors defined in Chapter 1. Indeed, ideas from quality control are used in creating some aspects of the models used. It is also important to note the major differences between inspection in the POE domain and in the quality control domain. In the latter the population of items has a mean and standard deviation that are attributed to a process, and the final product has specification limits that are applied to define acceptable and unacceptable items, thereby creating two groups. For the POE problem the overall container population is characterized by a mean and standard deviation that cannot be affected or influenced and consists of two distinct populations that the inspection aims to identify. The POE system of stations presents a
unique problem where sensor threshold levels and station sequence are not specified and
the optimal values are affected by many competing factors, whereas in classic quality
control the inspection limit is based on the specifications and process capability.

In this dissertation we focus our investigation on interesting problems unique to POE
inspection, beginning with the optimization of the inspection sequence and determination
of the sensors’ optimum threshold levels under different objectives and constraints in
systems that use a Boolean system decision function. We next consider the time spent in
inspection as an objective and although some researchers might account for the
competing objectives by introducing constraints such as budget or required throughout,
we include the terms directly in the objective function so that the true optimal system
performance can be found. We then investigate the impact of measurement errors on the
system performance and approaches that minimize this impact, including repeat
inspection and tightening the sensors’ threshold values.

In the problem of inspection allocation and scheduling defined in Chapter 6, the
scheduling of inspection operations bears similarities to the open shop scheduling
problem (OSSP) because there is no predefined sequence of operations. Details of
related OSSP research are provided in Chapter 6. Though optimal solutions are known
for some specified problem definitions (Brucker and Knust 2009), the problems with a
tardiness objective for which efficient solution methods have been found are limited—see
Liu and Bulfin (1988) and Timkovsky (2003). In addition, we consider a more
complicated problem in which the operations to be carried out are not predefined, and the
risk of falsely accepting a container is composed of multiple independent risk categories. The allocation of inspection operations affects the false acceptance rate and the schedule; therefore we present a mathematical programming formulation to simultaneously determine the inspection allocation and schedule that minimizes container tardiness and the probability of false acceptance.
CHAPTER 3

SEQUENCE AND THRESHOLD LEVEL OPTIMIZATION

The port-of-entry inspection problem is decomposed into two sub-problems. One problem deals with the determination of the optimum sequence of inspection or the structure of the inspection decision tree in order to achieve the minimum expected inspection cost. This problem can be formulated and investigated using approaches parallel to those used in the optimal sequential inspection procedure for reliability systems as described in Chapter 2. The other problem deals with the determination of the optimum threshold levels of sensors at inspection stations so as to minimize the cost associated with the probabilities of incorrectly accepting or rejecting a container. In this chapter we give an overview of the solution to the first (sequence) problem and apply those results in obtaining an overall inspection policy solution. Several optimization approaches are presented for determining the optimum sensor threshold levels while considering misclassification errors, total cost of inspection, and budget constraints. In contrast to previous work which determines threshold levels and sequence separately, they are considered as an integrated system and determined simultaneously.

This chapter is organized as follows. Section 3.1 describes the port-of-entry container inspection problem. Section 3.2 develops a procedure to find the optimal solution minimizing the cost of misclassifications and another procedure to minimize the cost of
inspection, and then combines the two approaches to minimize total cost. This section also introduces the receiver operating characteristic (ROC) curve to illustrate the inherent trade-off in the optimization problem of container classification and this work is also extended to include optimization under budget constraints. Section 3.3 presents numerical examples of the methods and finally the last section offers a discussion of the work presented.

3.1 Problem Description

3.1.1 Port-of-Entry Container Inspection System

The inspection of a container at a port-of-entry is performed sequentially at stations that form an inspection system. Containers are inspected and classified according to observations made regarding their attributes. Suppose there are \( n \) inspection stations in the system; one sensor (equipment) at each station is used to identify one attribute of the container being inspected, for example presence of radiation or uncharacteristic X-ray readings. One reading is taken and compared to a pre-specified threshold level to make a pass or fail decision for that particular attribute.

There are several categories into which we seek to classify the containers. In the simplest case, these are negative and positive, 0 or 1, with “0” designating containers that are considered “acceptable” and “1” designating containers that raise suspicion and require special treatment. After each inspection, we either classify the container as acceptable or subject it to another inspection process.
The classification is thought of as a system decision function $F$ that assigns to each binary string of decisions $(d_1, d_2, \ldots, d_n)$ a category. In this chapter, we focus on the case where there are only two categories. In other words $F(d_1, d_2, \ldots, d_n) = 0$ indicates negative class and that there is no suspicion with the container and $F(d_1, d_2, \ldots, d_n) = 1$ indicates positive class and that additional inspection is required, usually manual inspection.

In this chapter we define, for instance, a system that uses a series Boolean function as applying a decision function $F$ that assigns the container class “1” if any of the individual decisions are fail, $d_i = 1$ for any station $i$. For example, $F(d_1, d_2, \ldots, d_n) = (d_1 \lor d_2 \lor \ldots d_n)$. Furthermore, a system that uses a parallel Boolean function is defined as applying a decision function $F$ that assigns the container the class “1” only if all of the individual decisions are fails. For example, $F(d_1, d_2, \ldots, d_n) = (d_1 \land d_2 \land \ldots d_n)$.

### 3.1.2 Sensor Performance and Inspection Threshold Levels

Consider a large group of containers as a population of interest, some of which are unacceptable and should be rejected. Let $\pi$ represent the portion of unacceptable containers, $0 \leq \pi < 1$, thus $100 \times \pi$ percent of the containers should ideally be rejected. For a container randomly selected from this population, let $x$ represent its true status, with $x = 1$ representing that it should be rejected and $x = 0$ representing that it should be
accepted. We have $P(x = 1) = \pi$ and $P(x = 0) = 1 - \pi$.

Suppose this container is placed for inspection in an inspection system consists of $k$ sensors or inspection stations. Let $r_i$ be its measurement taken by the $i$th sensor, $i = 1, 2, \ldots, k$. Generally speaking, this measurement $r_i$ can be either numerical (continuous or discrete) or even graphical, as described in the Introduction section. To simplify the presentation of our development, however, we assume that $r_i$ is numerical and normally distributed. We choose to use the normal distribution because normally distributed data are the most commonly seen data in practice and it has been used in port-of-entry inspection applications (Stroud and Saeger 2003, Anand et al. 2006, Boros et al. 2009). Continuous measurements can often be transformed into normally distributed data by the well known inverse transformation method (Devroye 1986, p.28). Likewise, discrete data sometimes can be well approximated by a normal distribution either by applying the central limit theorem or by some special techniques such as variance stabilization transformation. For instance, as in the example involving particle count in the Introduction, $r_i$ can be a Poisson count. In this case, its square root transformation $\sqrt{r_i}$ is approximately normal distributed and has often been used in applications (Kihlberg et al. 1972). Lastly, our formulation can be extended to use any distribution with a calculable cumulative distribution function; we use the normal model here to illustrate our development.

In addition to the normal assumption, we expect the distribution of the measurement $r_i$ to
vary depending on the true status of the container, either $x = 1$ or $x = 0$. To reflect this relationship, we assume in particular that $r_i \sim N(\mu_{0i}, \sigma_{0i}^2)$ when $x = 0$ and $r_i \sim N(\mu_{1i}, \sigma_{1i}^2)$ when $x = 1$, with $\mu_{0i} \neq \mu_{1i}$. Here $(\mu_{0i}, \mu_{1i})$ and $(\sigma_{0i}, \sigma_{1i})$ are the mean and standard deviation parameters of the two normal distributions, respectively, and they are assumed to be known or can be estimated from past inspection history. In the port-of-entry inspection problem, the goal is to distinguish between the two groups of containers with $x = 0$ and $x = 1$. Thus, the relative distance between $\mu_{0i}$ and $\mu_{1i}$, but not their locations or scales, plays a key role in separating good and bad containers. Indeed, the problem is location and scale invariant (analogous to the problem of telling two temperatures apart without being affected by whether we use Celsius or Fahrenheit metric). Without loss of generality, we can simplify our notations by assuming $\mu_{0i} = 0$ and $\mu_{1i} = 1$. Otherwise, we set $r_i^{(\text{new})} = (r_i - \mu_{0i})/((\mu_{1i} - \mu_{0i}))$, $\sigma_{0i}^{(\text{new})} = \sigma_{0i}/((\mu_{1i} - \mu_{0i}))$ and $\sigma_{1i}^{(\text{new})} = \sigma_{1i}/((\mu_{1i} - \mu_{0i}))$, so we have $r_i^{(\text{new})} \sim N(0, \sigma_{0i}^2)$ when $x = 0$ and $r_i^{(\text{new})} \sim N(1, \sigma_{1i}^2)$ when $x = 1$.

We consider in this chapter the threshold approach. In this approach, each measurement $r_i$ is compared against a given threshold value $T_i$. The $i^{th}$ station rejects this item ($d_i = 1$) if the reading $r_i$ is higher than $T_i$ and accepts it ($d_i = 0$) if the reading is less than $T_i$. In this decision making process at the station level, there are two types of potential errors. For a randomly selected container with status $x = 0$, there is a chance
that a decision \( d_i = 1 \) is made at the \( i^{th} \) inspection. This is a Type I error of falsely rejecting a good item \( x = 0 \) and the probability of committing such an error can be computed by
\[
P(d_i = 1 | x = 0) = P(r_i > T_i | x = 0) = 1 - \Phi \left( \frac{T_i}{\sigma_{l_i}} \right).
\]
Also, for a container with true status \( x = 1 \), there is a chance that a decision \( d_i = 0 \) is made. This is a Type II error of falsely accepting a bad item \( x = 1 \). The probability of committing such an error is
\[
P(d_i = 0 | x = 1) = P(r_i \leq T_i | x = 1) = \Phi \left( \frac{T_i - 1}{\sigma_{u_i}} \right).
\]

### 3.1.3 System Inspection Policy

We consider the effect of inspection system parameters on the costs associated with performing inspection and misclassification of containers. It becomes clear that the overall performance of the inspection system is determined by both the sequence in which inspection stations are visited and the threshold levels applied at those stations, which we denote collectively as the inspection policy. Therefore the goal is to formulate the expected cost of inspection and classification errors (false positive and false negative) and use this information to generate a policy for the system’s optimum operation. It is assumed that the decision function is predetermined, while the sequence of stations visited and the sensor thresholds are decision variables of the optimization. The optimization method is illustrated for inspection systems with decision functions of series, parallel, series-parallel, and parallel-series Boolean functions. In principle the optimization framework in this chapter can be applied in considering the problem without
the assumption of a predetermined decision function.

3.2 Optimization Approaches

3.2.1 Minimization of Cost of Misclassification

At the system level, there are also two types of misclassification errors: falsely reject a container that should be cleared and falsely accept a container that should be rejected. The probability of false reject (PFR) is defined as the probability of the overall system rejecting a container conditional on the true status being acceptable. The probability of false accept (PFA) is defined as the probability of the system accepting a container conditional on the true status being unacceptable. The complementary probabilities of these two errors are true reject (PTR) and true accept (PTA). Denote by $D$ the decision of the entire inspection system of sensors where $D = 1$ means to reject, and $D = 0$ to accept. The four probabilities are listed as follows:

$$PFR = P(D = 1 | x = 0)$$  \hfill (3.1)

$$PTA = P(D = 0 | x = 0) = 1 - PFR$$  \hfill (3.2)

$$PFA = P(D = 0 | x = 1)$$  \hfill (3.3)

and

$$PTR = P(D = 1 | x = 1) = 1 - PFA.$$  \hfill (3.4)

The inspection decision of a system $D$ depends on the inspection results of its sensors and the system Boolean function. Some examples from Elsayed (1996) are given below.
Example 1. Series System

In a system using a series Boolean decision function, if any one station returns a fail decision, the overall container fails immediately. The PFR and PFA for the k-series system are given by:

\[
PFR^{[k]}_{series} = 1 - \prod_{j=1}^{k} P(d_j = 0 \mid x = 0) = 1 - \prod_{j=1}^{k} \Phi \left( \frac{T_j}{\sigma_{0j}} \right)
\]

and

\[
PFA^{[k]}_{series} = \prod_{i=1}^{k} P(d_i = 0 \mid x = 1) = \prod_{i=1}^{k} \Phi \left( \frac{T_i - 1}{\sigma_{1i}} \right)
\]

Example 2. Parallel System

In a system using a parallel Boolean decision function, if any one station returns a pass decision, the overall container is passed immediately. Thus, a container must fail every station to fail overall. The PFR and PFA for the k-parallel system are given by:

\[
PFR^{[k]}_{parallel} = \prod_{j=1}^{k} P(d_j = 1 \mid x = 0) = \prod_{j=1}^{k} \left[ 1 - \Phi \left( \frac{T_j}{\sigma_{0j}} \right) \right]
\]

and

\[
PFA^{[k]}_{parallel} = 1 - \prod_{i=1}^{k} P(d_i = 1 \mid x = 1) = 1 - \prod_{i=1}^{k} \left[ 1 - \Phi \left( \frac{T_i - 1}{\sigma_{1i}} \right) \right].
\]

Example 3. Parallel-Series System

In a system using a parallel-series Boolean decision function the container will fail overall only by failing at least one station in every parallel path. The PFR and PFA for
the \((n, m)\) parallel-series system (shown in Fig. 3.1) are given by:

\[
PFR_{\text{parallel-series}}^{[n,m]} = \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} P(d_{ij} = 0 | x = 0) \right] = \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} \Phi \left( \frac{T_{ij}}{\sigma_{0ij}} \right) \right]
\]  

(3.9)

and

\[
PFA_{\text{parallel-series}}^{[n,m]} = 1 - \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} P(d_{ij} = 0 | x = 1) \right] = 1 - \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} \Phi \left( \frac{T_{ij} - 1}{\sigma_{1ij}} \right) \right]
\]  

(3.10)

Figure 3.1. Conceptual Depiction of Parallel-Series System

**Example 4. Series-Parallel System**

In a system using a series-parallel Boolean decision function the container will fail overall only by failing every station within at least one subsystem. The \(PFR\) and \(PFA\) for the \((n, m)\) series-parallel system (shown in Fig. 3.2) are given by:
\[ PFR_{\text{series-parallel}}^{[n,m]} = 1 - \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} P \left( d_{ij} = 1 \mid x = 0 \right) \right] \]
\[ = 1 - \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} \left[ 1 - \Phi \left( \frac{T_{ij}}{\sigma_{0ij}} \right) \right] \right] \]  
(3.11)

and

\[ PFA_{\text{series-parallel}}^{[n,m]} = \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} P \left( d_{ij} = 1 \mid x = 1 \right) \right] \]
\[ = \prod_{i=1}^{n} \left[ 1 - \prod_{j=1}^{m} \left[ 1 - \Phi \left( \frac{T_{ij} - 1}{\sigma_{tij}} \right) \right] \right] \]  
(3.12)

Figure 3.2. Conceptual Depiction of Series-Parallel System

It should be noted that for parallel-series and series-parallel systems in which the number of sensors per path is not constant \((m\) varies by path), or the number of sensors per subsystem is not constant, the concepts of equations in Ex. 3 and 4 would hold true, with slight changes in notation. In general this extension is true wherever the parallel-series and series-parallel systems are discussed in this chapter.
Ideally, we seek a set of threshold values for the sensors and a system configuration for which both these errors are minimized. Unfortunately, this simultaneous minimization of both \( PFR \) and \( PFA \) is not possible, and reducing one error is likely to increase the other. Therefore, we need to define an optimal policy that balances such a trade-off.

One feasible approach is defining the expected cost of the system misclassifying a container. Let \( c_{FA} \) be the cost of the system accepting a “bad” container and \( c_{FR} \) be the cost of the system rejecting a “good” container. Then the total cost of system classification error is

\[
C_F = \pi \, PFA \, c_{FA} + (1 - \pi) \, PFR \, c_{FR}.
\]

The set of optimal threshold values is the one that minimizes the expected cost \( C_F \) of the system misclassifying containers over all possible combinations of sensors in the system and all possible threshold values:

\[
\{T_1, T_2, \ldots, T_k\} = \arg \min C_F. \quad (3.13)
\]

Currently this optimization problem is solved by complete enumeration of the possible combinations of discretized threshold level values and computation of \( C_F \). As the number of inspection stations increases the computational requirements increase significantly, rendering the complete enumeration approach impractical in some cases. Alternative approaches and heuristics should be considered.

The cost of false rejection is the cost of additional tests. In the practice of port-of-entry
inspection, these additional tests mean inspecting the contents manually. This is quite expensive since it might involve several workers for several hours, causing delays in completing the inspection and reduction in the inspection system throughput. Measuring the costs of false acceptance $c_{FA}$ is even more challenging. Indeed, the cost of missing a container that contains illegal drugs is not comparable to the cost of missing a container that holds a “dirty bomb”. One way is to assign a large cost value, say a few hundred -or even more- times the cost of a false reject.

An alternative and more flexible approach that avoids assigning exact values to the misclassification costs is the commonly used Receiver Operating Characteristics (ROC) curve method. The ROC analysis was first developed in psychophysics to summarize data from signal detection experiments (Green and Swets 1966). It has since been widely used in medicine, psychology, radiology, bio engineering, machine learning and data mining, among others. Depending on the practice, there are variations on how to produce an ROC curve, but all have the common feature of displaying two competing risks graphically as the parameter and condition changes. The most commonly seen ROC curves are in binary classifier systems, where the curve plots sensitivity values against (1 – specificity) values. Here, sensitivity refers to the probability of classifying an item positive when it is indeed positive and specificity is the probability of classifying an item negative when it is indeed negative.

In the current context of port-of-entry inspection, an ROC curve plots the probability of
true reject \( (PTR = 1 - PFA) \) against the probability of false reject \( (PFR) \) while varying the threshold parameters and the sequence of sensors. This kind of ROC curve provides a graphical representation of the trade-off between the probabilities of false accept (FA) and false reject (FR). It is a flexible and useful tool in decision-making (Armitage and Colton 1998).

Fig. 3.3 is an example of an ROC curve produced from a three sensor parallel system in the port-of-entry problem. Each point in the plots (a) or (b) represents a pair of \( (PFR, PTR) \) values for a given set of threshold level values and a specific sequence of the sensors. The most upper-left points form a curve which is referred to as the ROC curve. Theoretically speaking, in extreme cases, the ROC curve passes through two points \((0, 0)\) and \((1, 1)\). For any point not on the ROC curve, we can find a point on the ROC curve whose \( PFR \) or \( PFA \) or both values are better than those of the point that is not on the ROC curve. Thus, the ROC curve is the optimal curve in the sense of Pareto optimization. The ROC curve consists of the best choices of threshold values under different preferences.
The ROC curve is closely related to the aforementioned optimization problem (3.13). In particular, for a given set of \((c_{FA}, c_{FR})\) values in the optimization problem, there is a point on the ROC curve that corresponds to it. The three broken parallel lines \(i\), \(ii\) and \(iii\) in Fig. 3.3a have a slope equal to \(\frac{(1-\pi)}{\pi} \times \frac{c_{FR}}{c_{FA}}\), and they can be used to depict the process of optimization in problem (3.13). As the parallel lines move from \(i\) to \(iii\), the corresponding total cost of misclassification \(C_F\) decreases for those points on the lines.

The solution to the optimal problem (3.13) corresponds to the point \(A\) on the ROC curve, at which the tangent line \(iii\) intercepts the ROC curve. The extreme point \((0, 0)\) on the ROC curve corresponds to \(c_{FA} = 0\) and \(c_{FR} = \infty\), where the classifier finds no positives (detects no alarms). In this case, it always classifies the negative cases correct but it classifies all positive cases wrong. The extreme point \((1, 1)\) on the other end corresponds to \(c_{FA} = \infty\) and \(c_{FR} = 0\), where all containers are classified as positive. So all positive cases are correctly classified but all negative cases are misclassified (i.e. it raises a false
alarm on each negative case).

The ROC curve can be utilized in the port-of-entry inspection problem to assist in decision making, typically by choosing an “operating” point (a fixed point) on the ROC curve. The goal is to find the best trade-off between failing to detect positives against raising false alarms under given conditions. An illustrative example follows. Suppose we want to set a small tolerance level for the false acceptance rate ($PFA$) and, among those parameters that satisfy this constraint, choose a set that minimizes the false rejection rate ($PFR$). The tolerance constraint on $PFA$ corresponds to the $PTR$ constrained above the horizontal line as illustrated in Fig. 3.3b with tolerance level at 10%. Among those points above the horizontal line, the one that has the smallest $PFR$ is at the intersection (point $B$) of the horizontal line and the ROC curve; see Fig. 3.3b. This point $B$ is the operating point of this problem that minimizes $PFR$ at about 27% while holding the $PFA$ smaller than the preset tolerance value. The set of threshold values and sensor sequence that correspond to this operating point is the solution for this constraint optimization problem. Note that this approach does not involve the selection of the cost values ($c_{FA}, c_{FR}$).

3.2.2 Optimal Sequence for Expected Inspection Cost

In addition to the cost of making false decisions, there is also the cost of inspection itself. There are several ways to calculate the cost of obtaining a sensor reading. One would be to break down the cost of obtaining a sensor reading into two components: unit variable
cost and fixed cost. The unit variable cost is just the cost of using the sensor to inspect one container, and the fixed cost is the cost of the purchase and deployment of the sensor itself. In many cases, the primary cost is the unit variable cost since many inspections are very labor intensive. The fixed cost is usually a constant and often does not contribute to the optimization functions, so for simplicity we disregard the fixed cost. Thus, the inspection cost is basically the expected cost of making observations for a container. Note that depending on the system configuration, a container may or may not be inspected by all sensors. The arrangement of the sensors is closely related to the overall inspection cost.

Denote \( p_i \) by
\[
p_i = P(d_i = 0) = \sum_{j=0}^{1} [P(d_i = 0 \mid x = j) P(x = j)] = (1 - \pi) \Phi \left( \frac{T_i}{\sigma_0} \right) + \pi \Phi \left( \frac{T_i - 1}{\sigma_i} \right)
\]
(3.14)
and let \( q_i = 1 - p_i \). They are functions of threshold values \( T_i \). Let \( c_i \) be the inspection cost of sensor \( i \). Zhang et al. (2006) proves the following theorem.

**Theorem 1:**

(a) For a series Boolean decision function, inspecting attributes \( i = 1, 2, \ldots, n \) in sequential order is optimum in that it minimizes the expected inspection cost if and only if: \( c_1 / q_1 \leq c_2 / q_2 \leq \ldots \leq c_n / q_n \) (condition 1a).
In this case, the expected inspection cost is given by $C_i = c_i + \sum_{i=2}^{n} \left[ \prod_{j=1}^{i-1} p_j \right] c_i$.

(b) For a parallel Boolean decision function, inspecting attributes $i = 1, 2, \ldots, n$ in sequential order is optimum in that it minimizes the expected inspection cost if and only if:

$$c_1 / p_1 \leq c_2 / p_2 \leq \ldots \leq c_n / p_n$$

(condition 1b).

In this case, the expected inspection cost is given by $C_i = c_i + \sum_{i=2}^{n} \left[ \prod_{j=1}^{i-1} q_j \right] c_i$.

We generalize these results to systems with arrangements of parallel-series and series-parallel sensors, given in Theorem 2.

**Theorem 2:**

(a) Consider a parallel-series decision function that has $n$ parallel paths with $m$ sensors in series in each path (see Fig. 3.1). If an inspection system with attributes $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$ arranged in parallel-series is optimal, it satisfies the following conditions: the inspection sequences of the series of sensors within each path are arranged in the order of $c_{i,1} / q_{i,1} \leq c_{i,2} / q_{i,2} \leq \ldots \leq c_{i,m} / q_{i,m}$, and the inspection sequence of parallel paths is arranged in the order of $C_1 / P_1 \leq C_2 / P_2 \leq \ldots \leq C_n / P_n$ (condition 2a). Here, $P_i$ and $C_i$ are the probability of acceptance of the $i^{th}$ path and the minimal inspection cost: $P_i = P(D_i = 0) = \prod_{j=1}^{m} p_{i,j}$ and
\[ C_i = c_{i1} + \sum_{j=2}^{m} \left[ \prod_{k=1}^{j-1} p_{ik} \right] \quad c_{ij} = c_{i1} + \sum_{j=2}^{m} c_{ij} \prod_{k=1}^{j-1} \left( 1 - \pi \right) \Phi \left( \frac{T_{ik}}{\sigma_{0ik}} \right) + \pi \Phi \left( \frac{T_{ik} - 1}{\sigma_{1ik}} \right) \].

In this case, the minimal inspection cost is

\[ C_i = C_1 + \sum_{i=2}^{n} \left[ \prod_{j=1}^{i-1} (1 - P_j) \right] \quad C_i = C_1 + \sum_{i=2}^{n} C_i \prod_{j=1}^{i-1} \left( 1 - \prod_{k=1}^{m} p_{jk} \right). \]

(b) Consider a series-parallel decision function that has \( n \) subsystems in series with \( m \) units in parallel in each subsystem (see Fig. 3.2). If an inspection system with attributes \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \) arranged in series-parallel is optimal, it satisfies the following conditions: the inspection sequences of the parallel sensors within each subsystem are arranged in the order of \( c_{i1} / p_{i1,2} \leq c_{i2} / p_{i1,2} \leq \ldots \leq c_{im} / p_{i1,m} \), and the inspection sequence of the subsystems is arranged in the order of \( C_1 / Q_1 \leq C_2 / Q_2 \leq \ldots \leq C_n / Q_n \) (condition 2b). Here, \( Q_i \) and \( C_i \) are the probability of rejection of the \( i \)th subsystem and the minimal inspection cost:

\[ Q_i = P(D_i = 1) = \prod_{j=1}^{m} (1 - p_{ij}) \text{ and } \]

\[ C_i = c_{i1} + \sum_{j=2}^{m} \left[ \prod_{k=1}^{j-1} q_{ik} \right] \quad c_{ij} = c_{i1} + \sum_{j=2}^{m} c_{ij} \prod_{k=1}^{j-1} \left( 1 - \pi \right) \left( 1 - \Phi \left( \frac{T_{ik}}{\sigma_{0ik}} \right) \right) + \pi \left( 1 - \Phi \left( \frac{T_{ik} - 1}{\sigma_{1ik}} \right) \right) \].

In this case, the minimal inspection cost is

\[ C_i = C_1 + \sum_{i=2}^{n} \left[ \prod_{j=1}^{i-1} P_j \right] C_i = C_1 + \sum_{i=2}^{n} C_i \prod_{j=1}^{i-1} \left( 1 - \prod_{k=1}^{m} (1 - p_{jk}) \right). \]

The optimal arrangement of sensors depends on the values of \( p \)'s and \( q \)'s, which are functions of the threshold values. Therefore, given a set of threshold values the optimum
sequence is the one that satisfies the constraints 1a, 1b, 2a, or 2b stated in the preceding theorems.

### 3.2.3 Minimization of Expected Total Cost

In some situations, it is conceivable that we may consider the combined cost of inspection and container misclassification. The total expected cost \( C_{\text{Total}} = C_I + C_F \) is calculated from the results of the previous sections. To facilitate the computation for different systems with a large number of sensors, we provide induction methods which calculate the cost (both \( C_I \) and \( C_F \)) in Appendix A. The optimization problem now becomes finding a set of threshold values \( \{T_1, T_2, \ldots, T_k\} \) that minimizes the total expected cost:

\[
\{T_1, T_2, \ldots, T_k\} = \arg \min_{k} C_{\text{Total}}
\]

among the sets of threshold values that satisfy the constraints in Section 3.2.2.

The optimal set of threshold values from this optimization may be different from those obtained from optimization by minimizing cost of misclassification errors or minimizing inspection cost. In the case of port-of-entry inspection, the optimal solution of the cost-combined optimization may be very close to the solution obtained by the former if \( C_I \) is much smaller than the cost of the system misclassifying a container. Note that in practice in the port-of-entry inspection, the cost of false positives is often the cost of additional testing, such as opening the container and manually inspecting its contents. This is quite
expensive since it might involve several workers for hours, delays in completing the
inspection, and reduction in the inspection system throughput as stated earlier in the
chapter. In comparison to routine inspection cost of unit testing such as neutron or
gamma emissions detection, this FR cost is relatively high. The FA cost would be even
greater, including a huge potential social or economic impact.

3.2.4 Optimization with Budget Consideration

At a given port-of-entry inspection station, the inspection practice is often constrained by
budget. It is not possible to open and manually inspect every container or every cargo,
which is by far the most accurate but an extremely costly inspection method. If the
budget allows, we may want to allow more containers to be manually inspected, which in
turn affects the sensor inspection process. For example, if the budget is large, it is
possible to set low threshold levels to increase the $PTR$ of the sensor system, and flag
more containers for further manual inspection.

In our formulation, the total budget in an inspection station covers both the initial cost of
the inspection system and additional manual inspection cost. Therefore, the budget is
defined by

\[
\text{budget} = C_I + C_{\text{manual}} = C_I + c_{\text{unpack}} \left[(1 - \pi)PFR + \pi PFR\right]
\]  

(3.16)

where $C_I$ is the cost of initial system inspection, and $c_{\text{unpack}}$ is the unit cost of additional
manual inspection (unpacking the container).
Under the budget constraint, we maximize the probability of properly classifying suspicious containers passing through the entire inspection system, including the sensor inspection system and manual inspections. So, the optimization problem can be described as

$$\{T_1, T_2, \ldots, T_k\} = \arg \max PTR$$
subject to: Budget < $B_0$

where $B_0$ is the maximum available budget for the inspection system, and $\{T_1, T_2, \ldots, T_k\}$ is selected from possible threshold level values. We can formulate the budget constraint optimization problem similarly for other considerations. For example, minimization of the cost of misclassification can be obtained by finding the argument of the minimum of $C_r$ defined in Section 3.2.1.

This optimization problem can be presented by a graphical technique similar to the ROC curve, especially when we want to investigate the impact of the budget constraint. For instance, it is informative to investigate the relation between the chance of missing a suspicious cargo or dirty bomb and the budget. So, we plot the $PTR = 1 - PFA$ against the total budget $B_0$ while varying the threshold values and the sequence of sensors in an inspection system; see Fig. 3.8 and Section 3.3 for further details. The most upper-left points form a curve. This curve consists of points corresponding to optimal threshold values and best combination of sensors at different budget levels.
3.3 System Analysis with Numerical Examples

This section includes examples of solutions to the combined optimization problem of total cost (inspection and misclassification). Whereas the ROC curve was introduced as a graphical illustration of the inherent trade-off between \(PTR\) and \(PFR\), the minimum cost problem (3.15) produces an exact solution. Numerical examples are provided for the Boolean functions parallel, series, parallel-series, and series-parallel. Graphs of the optimization results are also presented. In addition, a numerical example is presented for the \(PTR\) maximization problem with budget consideration discussed in Section 3.2.4.

Fig. 3.4 presents the minimum total cost of an inspection policy given the following system information: parallel Boolean decision function, unit misclassification penalty costs \(c_{FR} = 500\) and \(c_{FA} = 100,000\), unit inspection cost \(c_1 = c_2 = c_3 = 1\), and distribution parameters \(\mu_{0i} = 0, \mu_{1i} = 1, \sigma_{0i} = 0.45, \sigma_{02} = 0.55, \sigma_{03} = 0.5, \text{ and } \sigma_{1i} = 0.5 \ (i=1,2,3)\). The results are arranged by varying \(T_1\) values along the horizontal axis and each point represents an optimal combination of threshold values and sequence, with the total cost along the vertical axis. The data series are the result of varying the parameter \(\pi\), the true portion of unacceptable containers.
Fig. 3.4 also illustrates that for a given set of parameters, there is an optimal sequence and threshold values that correspond to the minimum total cost. For example, given $\pi = 0.0002$, the combination of threshold values $T^* = \{T_1, T_2, T_3\} = [0.55, 0.45, 0.45]$ and the inspection sequence 3-1-2 results in the minimum total cost for a system implementing a parallel Boolean decision function. The variation of the $\pi$ value can influence the optimal inspection sequence and the optimal threshold values.
Fig. 3.5 presents the results for a series Boolean decision function. The parameter values and presentation of results are the same as in the parallel Boolean example. In comparison to the results for the parallel Boolean decision function, the curves for various $\pi$ values demonstrate significant overlap to the degree that the distinction between series is not apparent. This overlap indicates that the inspection sequence and threshold values for the series Boolean are not sensitive to the small $\pi$ values in the example. In general, the total cost of the series system is higher than that of parallel for the given parameters. This is due to the relative increase in the expectation of a false rejection.
Fig. 3.6 presents the results for a parallel-series Boolean decision function of two subsystems in parallel, each consisting of two sensors in series. Table 3.1 presents the sigma values for each of the four sensors in this example. For all sensors $\mu_0 = 0$ and $\mu_1 = 1$. The results are presented similarly to Figs. 3.4 and 3.5, with $T_1$ values across the horizontal axis and series resulting from various $\pi$ values.
Table 3.1 Distribution Parameter Values for Sensors in Parallel-Series Example

<table>
<thead>
<tr>
<th>(Subsystem, Sensor)</th>
<th>$\sigma_0$</th>
<th>$\sigma_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>0.25</td>
<td>0.35</td>
</tr>
<tr>
<td>(1,2)</td>
<td>0.65</td>
<td>0.25</td>
</tr>
<tr>
<td>(2,1)</td>
<td>0.45</td>
<td>0.55</td>
</tr>
<tr>
<td>(2,2)</td>
<td>0.55</td>
<td>0.35</td>
</tr>
</tbody>
</table>

Figure 3.7. Minimum Cost Curves for Series-Parallel System

Fig. 3.7 presents the results for a series-parallel Boolean decision function. The sensor distribution parameters are the same as in the previous parallel-series example, presented in Table 3.1.
Figure 3.8. Budget Curve for Parallel System

With the same parameters in the series system, we plot the relationship between the budget level and the probability of true reject; see Fig. 3.8. If $B_0 = 1.35$ on the horizontal axis, the optimal threshold values are $T^* = \{0.75 \ 0.05 \ 0.75\}$ and the probability of missing a suspicious cargo in the optimal case is about $1 - 0.9701 \approx 0.03$. If we increase the budget from $B_0 = 1.35$ to $B_0 = 1.50$, the optimal $T^* = \{0.5 \ 0.4 \ 0.6\}$ and the probability of missing a suspicious cargo decreases to 0.0002. Clearly, the 11% increase of the budget results in a significant increase in the detection of unacceptable containers. Now with $B_0 = 1.50$, an additional budget increase of the same amount results in little change in the probability of missing a suspicious cargo. It is not cost effective to apply
the additional amount of inspection. Such information may help decision makers in assigning appropriate budgets to the port-of-entry inspection stations.

3.4 Discussion

In this chapter we investigate the port-of-entry problem with a small number of inspection stations. Complete enumeration of all possible threshold levels for each sensor resulted in determining the optimum threshold levels for the sensors such that the total cost is minimized. This has been done for sensors arranged in series, parallel, parallel-series, and series-parallel configurations. The key factor that has a direct effect on the determination of the sensors threshold levels is the cost of misclassification of a container with Type I error. This cost is difficult to estimate as it is a function of many unknowns but the effect could be catastrophic. Clearly, tightening the threshold levels minimizes the Type I error but may increase the cost of delaying the container. This has not been considered in this research but raises an important consideration of not only the cost of container misclassification but also the cost of delay incurred in the system.

Hence, we have two conflicting objectives, which lead to a multi-objective optimization problem, which we will investigate in Chapter 4. Likewise, the optimum inspection sequencing problem in a multi-objective problem will be addressed. The effect of measurement error on system performance and sensitivity analysis of system parameters present other areas for future research and will be discussed in Chapter 5. Finally, the determination of the optimum threshold levels of sensors arranged in non standard
arrangements such as a general network of sensors or $k$-out-of-$n$ arrangement warrants further investigation.

Another point of interest is that in this chapter we use a broad definition of containers considered unacceptable by border inspection authorities for various reasons. This is a basic assumption of the current model, assuming one $\pi$ value to reflect all containers having any unacceptable contents. This model can be extended to consider individual types of risk with a different $\pi_i$ value for each type. This extension would have significant effects which must be incorporated into the construction of the model. To include more individual assumptions of risks would require more historical data to generate information and distributions; however more specific misclassification cost values associated with the various risks could be used. Chapter 6 investigates the use of $\pi_i$ and cost values for separate risks.
CHAPTER 4

MULTI-OBJECTIVE OPTIMIZATION

This chapter extends the work in Chapter 3 by taking into consideration the time required to complete inspection and treating it as another objective to be minimized. As discussed in the Introduction, the delay incurred by an inspection system is a major concern as it has significant economic consequences. A new formulation is developed for the POE problem by considering both the minimization of the total cost and the delay time of containers simultaneously as a multi-objective optimization problem. The theorems for finding the optimal sequence of inspections for expected cost are modified to find an optimal sequence that considers cost and time simultaneously. Similar to the problem investigated in Chapter 3, we seek the optimal inspection sequence and the optimal threshold levels of sensors at inspection stations in order to minimize the total cost and total delay time. Main problem assumptions involving the inspection system, Boolean decision function, sensor performance, and formulation of error probabilities remain as stated in Chapter 3. In this chapter we assume that the time required for inspection at a station to be related to the inspection policy parameters.

This chapter is organized as follows. Section 4.1 describes the multiple objectives of the optimization problem: the cost of misclassifications, the cost of inspection, and the time spent in inspection. Section 4.2 details approaches for solving the multi-objective
4.1 Performance Measures of Inspection Policy

The problem description, assumptions, and notations detailed in Sections 3.1.1 and 3.1.2 of Chapter 3 are applicable here. The minimization of costs associated with performing inspection and misclassification of containers has been formulated in Chapter 3. Here we expand the optimization objective to include the time required for inspection, which takes into account the effect of delays on the overall system. The performance of the inspection system is determined by both the sequence in which inspection stations are visited and the threshold levels applied at those stations, which we denote collectively as the inspection policy.

Since the optimal parameter values for the cost minimization problem may not minimize time, some compromise may be required. A particular balance of the importance of cost and time may be represented by weights. We consider the case where the relative importance of cost and time is unspecified and therefore we use different importance weights to generate possible solutions that produce a Pareto frontier as described in Section 4.2.

4.1.1 Cost of Misclassification and Inspection

The cost involved in this inspection problem is the sum of any cost incurred as a result of
misclassifying a container’s status and the actual cost of performing the inspection. There are two types of misclassification errors at the systems level: falsely rejecting a container that should be cleared and falsely accepting a container that should be rejected. The four probabilities of false and true classification outcomes are $P_{FR}$, $P_{FA}$, $P_{TR}$, and $P_{TA}$. These probabilities are defined in Chapter 3 and calculations are provided for four example Boolean decision functions, however the probability equations can be rewritten in terms of the threshold value $T_i$ and $\sigma$ values related to the inspection station for any given Boolean function. The total cost of system classification error is defined in Chapter 3 as:

\[
C_r = \pi P_{FA} c_{FA} + (1 - \pi) P_{FR} c_{FR},
\]

(4.1)

where $c_{FA}$ is the cost of the system accepting an unacceptable container and $c_{FR}$ is the cost of the system rejecting an acceptable container.

The expectation of the cost of inspection $C_I$ is a function of the unit cost to operate each sensor at a station and the probabilities of passing each station. Given a particular set of threshold values, an optimal sequence (minimizing the cost of inspection) in which to visit the stations can be found following the conditions stated in Chapter 3. The total cost per container arising from misclassification errors and inspection is denoted by $c_{total} = C_r + C_I$. 
4.1.2 Time for Inspection

The time required for a container to pass through an inspection station is an important measure of the inspection system performance. It is possible that this time would be related to some characteristic of the inspection station, that is to say the inspection may be sped up or slowed down depending on some operational setting of the sensor. For example the inspection time may be related to a variable that represents the resolution or other settings of the sensor. Jupp et al. (2000) report that high statistical profiles are obtained by collecting data for 180 second at each position, i.e., time required to integrate energy and generate profiles for detecting explosives is indirectly related to the threshold level. Therefore, we assume the time \( t_i \) spent at a station could be related to the threshold level \( T_i \) at that station. For illustrative purposes we assume in our numerical examples later in Sections 4.2.2 and 4.3 the relationship could be an exponential function, expressed as \( t_i = a \exp(bT_i) \). Other expressions could also have been used.

To find the total expectation of time spent in the system for a given container we first denote \( p_i \), the probability of passing station \( i \), by:

\[
p_i = P(d_i = 0) = \sum_{j=0}^{1} [P(d_i = 0 \mid x = j)P(x = j)] = (1 - \pi) \Phi \left( \frac{T_i}{\sigma_{0i}} \right) + \pi \Phi \left( \frac{T_i - 1}{\sigma_{1i}} \right) \tag{4.2}
\]

and \( q_i = 1 - p_i \). Clearly \( p_i \) and \( q_i \) are functions of the threshold value \( T_i \). Then, the total expected inspection time per container \( t_{\text{total}} \) for a system with \( n \) stations using a series Boolean decision function can be expressed as:
\[ t_{\text{total}} = t_1 + \sum_{i=2}^{n} \left( \prod_{j=1}^{i-1} p_{ij} \right) t_i, \]  

(4.3)

where \( t_i \) is the inspection time at station \( i \). For a parallel Boolean decision function, the total expected inspection time per container is:

\[ t_{\text{total}} = t_1 + \sum_{i=2}^{n} \left( \prod_{j=1}^{i-1} q_{ij} \right) t_i. \]  

(4.4)

4.2 Multi-Objective Optimization

4.2.1 Total Expected Cost and Time

As noted in the problem introduction, we need to determine the optimal design or configuration of sensors in the system and the optimum sets of threshold levels that can achieve the objectives of maximizing inspection system throughput and minimizing the expected total cost. This is a typical multi-objective optimization problem. See, for instance, Eschenauer et al. (1990), Statnikov and Matusov (1995), Fonseca and Fleming (1998a, 1998b), and Leung and Wang (2000), among others. We formulate the POE problems as a multi-objective optimization problem:

\[ \min_{\text{Sequence,Threshold}} \left\{ c_{\text{total}}, t_{\text{total}} \right\} \]  

(4.5)

In general, there may be a large number or infinite number of optimal solutions in the sense of Pareto-optimality. It is desirable to find as many (optimal) solutions as possible in order to provide more choices to decision makers.
The multi-objective problem is almost always solved by combining the multiple objectives into one scalar objective whose solution is one of the Pareto optimal points for the original problem. A commonly used method to deal with the multi-objective optimization problem is to use the weighted sum approach, where we optimize fitness functions such as $f_{w_1,w_2}(S,T) = w_1c_{total} + w_2t_{total}$ (i.e., weighted sums of the objective functions) for various choices of fixed weights $w_1$ and $w_2$, $w_1 + w_2 = 1$. Here, $S$ and $T$ stand for sequence and threshold levels. Thus, the multi-objective optimization problem becomes a sequence of single objective optimization problems, in which we minimize the fitness function for a set of fixed weights $w_1$ and $w_2$:

$$\min_{S,T} f_{w_1,w_2}(S,T)$$  \hspace{1cm} (4.6)

In this chapter, we employ a modified weighted sum approach, in which we utilize some theoretical results to deal with the arrangement of system sequences. Note that the function $f_{w_1,w_2}(S,T)$ is very sensitive due to the discrete nature of the station sequence, in that a switch in the sequence may result in a significant change in the function. The number of arrangements also grows exponentially as the number of inspection stations increase. It is computationally expensive to directly solve the minimization problem in Eq. (4.6). For the system Boolean functions considered in this chapter, the optimal sequence can be obtained for a given set of weights and thresholds, as stated in the theorem presented below. So, for a given set of weights and threshold we are able to compute the function:
\[ f_{w_1, w_2}(T) = \min_S f_{w_1, w_2}(S, T) \] (4.7)

without using an optimization algorithm. In the modified weighted sum approach, we in fact solve the minimization problem \( \min_T f_{w_1, w_2}(T) \).

This modified approach can provide an efficient method to deal with the multi-objective optimization problem under the current context by applying a theorem to select the optimal sequence out of all possible and thus limiting the search area.

**Theorem 1:**

(a) For a series Boolean decision function, inspecting attributes \( i = 1, 2, \ldots, n \) in sequential order is optimum, in the sense of minimizing the fitness function for the given set of weights \( (w_1, w_2) \) and a given set of thresholds, if and only if:

\[
(w_1 c_i + w_2 t_i)/q_i \leq (w_1 c_2 + w_2 t_2)/q_2 \leq \ldots \leq (w_1 c_n + w_2 t_n)/q_n \quad \text{(condition 1a)}. \]

In this case, the minimal value of the fitness function is given by

\[
f_{w_1, w_2}(T) = (w_1 c_1 + w_2 t_1) + \sum_{i=2}^{n} \left[ \prod_{j=1}^{i-1} p_j \right] (w_1 c_i + w_2 t_i) + w_1 C_F. \]

(b) For a parallel Boolean decision function, inspecting attributes \( i = 1, 2, \ldots, n \) in sequential order is optimum, in the sense of minimizing the fitness function for the given set of weights \( (w_1, w_2) \) and a given set of thresholds, if and only if:

\[
(w_1 c_1 + w_2 t_1)/p_1 \leq (w_1 c_2 + w_2 t_2)/p_2 \leq \ldots \leq (w_1 c_n + w_2 t_n)/p_n \quad \text{(condition 1b)}. \]
In this case, the minimal value of the fitness function is given by

\[ f_{w_1, w_2}(T) = (w_1c_1 + w_2t_1) + \sum_{i=2}^{n} \left( \prod_{j=1}^{i-1} q_{ij} \right) (w_1c_i + w_2t_i) + w_1C_F. \]

The results in Theorem 1 for series system and parallel system can be extended to systems using parallel-series and series-parallel decision functions, given in Theorem 2.

A parallel-series decision function might be useful if each path represents an indicator of one particular risk and a fail decision in every path signifies the presence of that risk. A series-parallel decision function might be useful if each subsystem in series represents a different risk and a fail decision in any subsystem is significant.

**Theorem 2:**

(a) Consider a parallel-series decision function that has \( n \) parallel paths with \( m \) sensors in series in each path. If an inspection system with attributes \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \) arranged in parallel-series is optimal, it satisfies the following conditions: the inspection sequence of the series of sensors within each path are arranged in the order of \((w_1c_{i1} + w_2t_{i1}) / q_{i1} \leq (w_1c_{i2} + w_2t_{i2}) / q_{i2} \leq \ldots \leq (w_1c_{im} + w_2t_{im}) / q_{im}\), and the inspection sequence of parallel paths is in the order of \( F_1 / P_1 \leq F_2 / P_2 \leq \ldots \leq F_n / P_n \) (condition 2a). Here, \( F_i \) and \( P_i \) are the minimal combined fitness of cost and time and the probability of acceptance of the \( i^{th} \) path: \( F_i = (w_1c_{i1} + w_2t_{i1}) + \sum_{j=2}^{m} \left( \prod_{k=1}^{j-1} p_{ik} \right) (w_1c_{ij} + w_2t_{ij}) \).
and \( P_i = P(D_i = 0) = \prod_{j=1}^{m} p_{ij} \).

In this case, the minimal value of the fitness function is:

\[
f_{w_1, w_2}(T) = F_1 + \sum_{i=2}^{n} \left[ \prod_{j=1}^{i-1} (1 - P_j) \right] F_i + w_i C_F = F_1 + \sum_{i=2}^{n} F_i \prod_{j=1}^{i-1} \left( 1 - \prod_{k=1}^{m} p_{jk} \right) + w_i C_F.
\]

(b) Consider a series-parallel decision function that has \( n \) subsystems in series with \( m \) sensors in parallel in each subsystem. If an inspection system with attributes \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, m \) arranged in series-parallel is optimal, it satisfies the following conditions: the inspection sequences within each subsystem are arranged in the order of \( (w_i c_{i1} + w_2 t_{i1})/p_{i1} \leq (w_i c_{i2} + w_2 t_{i2})/p_{i2} \leq \ldots \leq (w_i c_{im} + w_2 t_{im})/p_{im} \) and the inspection sequence of the series of subsystems is in the order of \( F_1/Q_1 \leq F_2/Q_2 \leq \ldots \leq F_n/Q_n \) (condition 2b). Here, \( F_i \) and \( Q_i \) are the minimal combined fitness of cost and time and the probability of rejection of the \( i \)th subsystem:

\[
F_i = (w_i c_{i1} + w_2 t_{i1}) + \sum_{j=2}^{m} \left[ \prod_{k=1}^{j-1} q_{ik} \right] (w_i c_{i j} + w_2 t_{ij}) \quad \text{and}
\]

\[
Q_i = P(D_i = 1) = \prod_{j=1}^{m} \left( 1 - p_{ij} \right). \quad \text{In this case, the minimal value of the fitness function is:}
\]

\[
f_{w_1, w_2}(T) = F_1 + \sum_{i=2}^{n} \left( \prod_{j=1}^{i-1} P_j \right) F_i + w_i C_F = F_1 + \sum_{i=2}^{n} F_i \prod_{j=1}^{i-1} \left( 1 - \prod_{k=1}^{m} (1 - p_{jk}) \right) + w_i C_F.
\]

Appendix B shows the proof of the theorems. From the theorems, we describe our modified weighted sum optimization algorithm as follows:

**Step 1:** Generate \( N \) sets of weight pairs \((w_1, w_2)\), where \( N \) is a large number;
Step 2: For each pair of weights, solve the minimization problem

\[ T_{\text{min}}^{(w_1,w_2)} = \arg \min_T f_{w_1,w_2}(T) \quad \text{where the function} \quad f_{w_1,w_2}(T) \quad \text{is computed in a subroutine stated below;} \]

Step 3: Obtain the optimal sequence corresponding to \( T_{\text{min}}^{(w_1,w_2)} \) (utilizing the results of the theorems) and compute the corresponding optimal throughput time and total cost \( (t_{\text{total}}^{(w_1,w_2)}, c_{\text{total}}^{(w_1,w_2)}) \);

Step 4: Plot the \( N \) pairs of optimal throughput time and cost \( (t_{\text{total}}^{(w_1,w_2)}, c_{\text{total}}^{(w_1,w_2)}) \) which form the Pareto optimal solutions for the multi-objective optimization problem.

For the parallel and series inspection Boolean systems, we use the following subroutine to calculate the function \( f_{w_1,w_2}(T) = \min_S f_{w_1,w_2}(S,T) \) in Step 3 of the weighted sum optimization algorithm:

1) For each sensor \( i \), calculate \( w_1c_i + w_2t_i \);

2) For each sensor \( i \), calculate the ordering criterion \( (w_1c_i + w_2t_i)/p_i \) or \( (w_1c_i + w_2t_i)/q_i \);

3) Sort the ordering criteria to find the optimal arrangement of sensors according to the theorems;

4) Calculate the total cost \( c_{\text{total}} \) and the expected time of inspection \( t_{\text{total}} \) and return

\[ f = w_1c_{\text{total}} + w_2t_{\text{total}}. \]

A similar subroutine can be developed for series-parallel and parallel-series Boolean...
4.2.2 Computing Approaches

Standard minimization techniques, such as Newton-Raphson type or golden section search and parabolic interpolation algorithms, could perform poorly here due to the discrete nature of the objective function, as discussed in Section 4.2.1. This is why the modified weighted-sum algorithm is proposed, and a program in MATLAB (The MathWorks 2008) is developed to implement it. In Step 2 of the algorithm, the minimization of the function $f_{w_1, w_2}(T)$ is carried out by the built-in MATLAB random search based `ga` function.

A genetic algorithm is an iterative random search algorithm which takes advantage of information in the previous steps (ancestors) to produce new searching points (offspring). It is called a “genetic” algorithm because the principle and design mimic those of genetic evolution found in nature (Mitchell 1996). A genetic algorithm can be applied to solve “a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear” (The MathWorks 2008).

In the optimization algorithm developed here, the MATLAB function `ga` was used to minimize $f_{w_1, w_2}(T)$ for each pair of weights. Note that the function $f_{w_1, w_2}(T) = \min_{S} f_{w_1, w_2}(S, T)$ is discrete with regards to $T$ which is inherited from the
sequence optimization. Since the optimization function is complex, we compared the results from this method against a complete enumeration method for verification.

We refer to the algorithm using the \( ga \) function as the GA approach. To verify the results, a grid search method (GS) is implemented. The grid search method does not use either of the developments (theorem or algorithm) in this chapter. It is a complete enumeration of possible threshold values and all inspection sequences. A discrete set of threshold values is formed in the range [0,1] using a gradient of 0.05. The total cost and total time are calculated for each possible combination of threshold values and sequence. The resulting cost and time values are plotted and the outermost points along the curve are filtered to represent the solution set that forms the Pareto frontier. Thus the GS method yields a small number of true optimal points compared to the GA method.

The results of the multi-objective optimization are presented in graphical form. The two graphs in Fig. 4.1 illustrate the optimal points obtained from the GS and GA methods applied to an inspection system using a parallel Boolean decision function. The system parameters in this example are as follows: \( n = 3, \ c = [1 \ 1 \ 1], \ \pi = 0.0002, \ \mu_0 = [0 \ 0 \ 0], \mu_i = [1 \ 1 \ 1], \ \sigma_0 = [0.16 \ 0.2 \ 0.22], \ \sigma_i = [0.3 \ 0.2 \ 0.26], \ c_{FA} = 100000, \ c_{FR} = 500, \ a = [20 \ 20 \ 20], \ b = [-3 \ -3 \ -3], \ w_1 = [0: 0.004:1], \ w_2 = 1 - w_i. \) Square brackets list three specific values corresponding to the three stations in the example. Note that the values of \( b \) cause the inspection time to decrease as the threshold level increases.
The grid search method produces optimal points that fall into distinct vertical segments due to the discrete nature of the method which only considers $T$ such that $T = 0.05m$, $m = 0, 1, 2, ... 20$. The minimum search gradient with an acceptable computation time was used. The left graph contains only the outermost points with respect to the Pareto frontier from this method. Note that only a small number of the points fall on the theoretical Pareto frontier.

The right graph illustrates the optimal points obtained from the GA method, which seem to include or improve upon the Pareto frontier of solutions with minimal time and cost.
from the GS method. Each point represents the time and cost for one possible solution, and each solution is defined by a set of threshold values \( \{ T_i : i = 1, \ldots, n \} \) - each to be applied at one of the \( n \) inspection stations - and the sequence in which to visit those stations. Table 4.1 presents three examples of points chosen from the Pareto frontier of solutions.

<table>
<thead>
<tr>
<th>( T_1 )</th>
<th>( T_2 )</th>
<th>( T_3 )</th>
<th>Sequence</th>
<th>Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.95</td>
<td>0.05</td>
<td>2-3-1</td>
<td>9.03</td>
<td>1.16</td>
</tr>
<tr>
<td>0.0</td>
<td>0.85</td>
<td>0.0</td>
<td>2-1-3</td>
<td>5.54</td>
<td>1.57</td>
</tr>
<tr>
<td>0.0</td>
<td>0.75</td>
<td>0.05</td>
<td>2-3-1</td>
<td>3.13</td>
<td>2.11</td>
</tr>
</tbody>
</table>

It is important to consider program running time in the comparison of methods. The GS method with grid of 0.05 runs in about 6 minutes, however only about 12 points of the output are estimated to fall within the theoretical Pareto frontier. If the grid is decreased to 0.025, roughly 23 points on the theoretical Pareto frontier are produced but it takes 5 hours to run. Further reducing the grid to 0.01 requires more than 200 hours to finish. Therefore it becomes impractical to decrease the grid size in order to generate more optimal points on the theoretical frontier.

The GA method takes about 10.5 hours with the current choice of parameters (PopulationSize=80) and produces 251 points on the theoretical Pareto frontier. Note that
the *ga* function of MATLAB is designed for general purpose use, and we anticipate that the running time can be significantly improved by using a specialized program. Moreover, the GA method produces optimal solutions in all trials that best represent the theoretical Pareto frontier.

The GA method is applied to a system using a series Boolean decision function with the same parameters as the first example. The results are presented in Fig. 4.2a. It is evident that a change in Boolean function has an effect on the results.

![Figure 4.2. GA Method Results](image)

In the third example the GA method is used to find the multi-objective optimal solution to an inspection problem that uses a series-parallel Boolean function with the system parameters: $m = 2$, $n = 2$, $c = [1 \ 1; 1 \ 1]$, $\pi = 0.0002$, $\mu_0 = [0 \ 0; 0 \ 0]$, $\mu_1 = [1 \ 1; 1 \ 1]$, $\sigma_0 = [0.16 \ 0.2; 0.22 \ 0.18]$, $\sigma_1 = [0.3 \ 0.2; 0.26 \ 0.18]$, $c_{FA} = 100000$, $c_{FR} = 500$, $a = [20 \ 20; 20 \ 20]$, $b = [-3 \ -3; -3 \ -3]$, $w_1 = [0: 0.004:1]$, $w_2 = 1 - w_1$. Fig. 4.2b gives the optimal
points for this example.

4.3 Numerical Example

In order to provide some design guidelines for system configuration, we carry out a design of experiment for a system using an $n = 3$ series Boolean decision function. The GA method is utilized in this example.

We specify the following parameters: $\pi$, $\mu_0$, $\mu_1$, $c$, $\sigma_0$, $\sigma_1$, $c_{FA}$, $c_{FR}$, $a$, and $b$. In the design of experiment, we fix $\mu_0 = [0 0 0]$, $\mu_1 = [1 1 1]$, $c = [1 1 1]$, $a = [5 5 5]$, and $b = [-0.8 -0.8 -0.8]$ and consider the four factors $\pi$, $\{\sigma_0, \sigma_1\}$, $c_{FA}$ and $c_{FR}$. Each factor has two or three levels as described below. The set $\{\sigma_0, \sigma_1\}$ is considered as one factor. We now describe the parameters in details.

$\pi$: probability of an unacceptable container

The probability of a container being unacceptable varies. We assume that two unacceptable containers per 10000 containers may be appropriate, and choose one fourth of this rate for comparison. Therefore, the two levels of $\pi$ are 5E-05 and 2E-04.

$\{\sigma_0, \sigma_1\}$: standard deviations of the measurements for acceptable and unacceptable containers, respectively
To define this factor we assume standard deviation values are among \{0.15 \ 0.25 \ 0.35\}. Furthermore we choose sets of \{\sigma_0, \sigma_1\} together to affect the area of overlapping probability of the two normal distributions of measurements for each of the three stations.

In the following analysis, this overlap is considered as a factor with three levels—small, moderate, and large. In particular the sets of \{\sigma_0, \sigma_1\} corresponding to the levels are:

- \{\sigma_0, \sigma_1\} = \{[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]\} small overlap,
- \{[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]\} moderate overlap, and
- \{[0.25 \ 0.25 \ 0.35], [0.35 \ 0.35 \ 0.25]\} large overlap.

\(c_{FA}\): cost of the system accepting an unacceptable container

Accepting an unacceptable container has more severe consequences than rejecting an acceptable container. Hence we use a much higher value for \(c_{FA}\) than \(c_{FR}\). We use 1E+05 as the first level of \(c_{FA}\) and 1E+07 as the second level.

\(c_{FR}\): cost of the system rejecting an acceptable container

Relative to the levels of \(c_{FA}\), we use 200 and 400 as two levels of \(c_{FR}\).

The full 24 (= 2×3×2×2) designs are listed in Appendix C. For each design, we use five weights: \((w_1, w_2) = (0,1), (0.25,0.75), (0.5,0.5), (0.75,0.25), (1,0)\) and use the genetic algorithm and modified weighted sum optimization algorithm in MATLAB to obtain the optimal threshold levels and optimal inspection sequences. As an example, the result of design 19 is described in Table 4.2.
Table 4.2. Optimal Solution of Design 19

<table>
<thead>
<tr>
<th>$w_1, w_2$</th>
<th>Cost</th>
<th>Time</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1</td>
<td>65.92</td>
<td>6.73</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3-2-1</td>
</tr>
<tr>
<td>0.25, 0.75</td>
<td>6.21</td>
<td>8.37</td>
<td>0.57</td>
<td>0.75</td>
<td>0.87</td>
<td>3-2-1</td>
</tr>
<tr>
<td>0.5, 0.5</td>
<td>5.65</td>
<td>8.63</td>
<td>0.47</td>
<td>0.71</td>
<td>0.91</td>
<td>3-2-1</td>
</tr>
<tr>
<td>0.75, 0.25</td>
<td>5.62</td>
<td>8.68</td>
<td>0.46</td>
<td>0.70</td>
<td>0.91</td>
<td>3-2-1</td>
</tr>
<tr>
<td>1, 0</td>
<td>5.61</td>
<td>8.71</td>
<td>0.46</td>
<td>0.70</td>
<td>0.91</td>
<td>3-2-1</td>
</tr>
</tbody>
</table>

Given a set of weights, our objective is to minimize $w_1 c_{total} + w_2 t_{total}$. Therefore, $(w_1, w_2) = (0, 1)$ corresponds to minimizing total time only and $(w_1, w_2) = (1, 0)$ gives the solution for minimizing total cost only. To gain a general understanding of the effects of four factors in our exploratory analysis, we produce boxplots and use analysis of variance models (ANOVA) to study the outcomes. Since the ranges of total cost and total time for the 24 designs are in similar scale (the range of cost is [3.329, 20.573] and the range of time is [6.730, 8.894]) for equal weights $(w_1, w_2) = (0.5, 0.5)$, we use these results for illustration in our analysis. The conclusions from analyses using other choices of the weights are similar.
Fig. 4.3 shows the boxplots of four factors. The boxplot of factor $\pi$ indicates that a higher probability of an unacceptable container results in higher cost and time (i.e., higher value of $0.5c_{total} + 0.5t_{total}$). The boxplots of $c_{FA}$ and $c_{FR}$ show that higher costs of false decision have higher cost and time. The boxplot of the level of overlap in $\{\sigma_0, \sigma_1\}$ illustrates that larger overlapping distributions result in higher cost and time. In order to see which factors have significant effect, we fit an analysis of variance model and list the results in Table 4.3. The table shows that the factors of $\{\sigma_0, \sigma_1\}$ and $c_{FA}$...
have significant effect under significance level 0.05, while \( \pi \) and \( c_{FR} \) do not. These conclusions match well with our intuition. Note that, with a higher level of overlap in the choice of \( \{\sigma_0, \sigma_1\} \), the probability of false decision may be greater. Also, higher costs of false decision increase the total cost.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr (&gt; F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi )</td>
<td>1</td>
<td>5.938</td>
<td>5.938</td>
<td>3.009</td>
<td>0.0999</td>
</tr>
<tr>
<td>( {\sigma_0, \sigma_1} )</td>
<td>2</td>
<td>35.66</td>
<td>17.83</td>
<td>9.036</td>
<td>0.0019</td>
</tr>
<tr>
<td>( c_{FA} )</td>
<td>1</td>
<td>52.135</td>
<td>52.135</td>
<td>26.422</td>
<td>6.86e-05</td>
</tr>
<tr>
<td>( c_{FR} )</td>
<td>1</td>
<td>2.51</td>
<td>2.51</td>
<td>1.272</td>
<td>0.2742</td>
</tr>
</tbody>
</table>

4.4 Discussion

This chapter investigates and formulates the inspection systems at ports-of-entry. It is formulated as a multi-objective optimization problem that attempts to minimize the total cost as well as the delay time of the container inspection. The formulation is general and applicable to different systems as the attributes of a typical container are expressed by a Boolean function. The inspection stations in the system can be arranged in series (sequential inspection), parallel, series-parallel, parallel-series, \( k \)-out-of-\( n \) (where any \( k \) stations out of \( n \) indicate the presence of undesired attributes) or in any network configuration. Boolean functions corresponding to any of these configurations can be
stated and subsequently an appropriate formulation can be expressed. The number of attributes and the inspection sequence have significant impact on the system performance. Likewise, the threshold levels of the sensors are critical in the decision process of accepting or classifying a container as suspicious. They influence the probability of making the “wrong” decision in accepting undesired containers or subjecting acceptable containers to further unneeded inspections.

Based on a numerical study, the proposed weighted sum approach with genetic algorithm is capable of determining the optimum inspection sequence and the threshold levels at each inspection station that result in the optimal system performance measures of cost and time. Another numerical study using design of experiments is carried out and illustrated in a system applying a series Boolean function. It provides a systematic way to study and identify factors that are important in the design of a system. Finally, the multi-objective optimization approach provides Pareto frontier optimal solutions where each solution consists of the optimum sequence of the inspection stations and the corresponding optimum threshold levels. This enables the decision maker to choose amongst solutions that meet other constraints such as budget, space or layout of the port. Besides the approach proposed in this chapter, the multi-objective problem could also be solved using the methods discussed in Chapter 5.

This research focuses on the optimization of an inspection system given current environment and parameters. There are some limitations, for instance it does not consider the intentions or behaviors of potential smugglers. The information regarding
the intention or behaviors of smugglers is often obtained from intelligence gatherings and from studies of past events. This type of information has an impact on the probability parameter $\pi$ and it could be incorporated in our model, for example the parameter $\pi$ can be adjusted or estimated based on perceived behavior of the smugglers, as well as other potential factors. Needless to say, POE inspection is a very important and very complex problem. What we have outlined here is part of research trying to provide guidelines to improve the effectiveness and efficiency of the practice of the POE inspection.
CHAPTER 5

MODELING MEASUREMENT ERROR

The previous chapters develop a model of the POE inspection system and illustrate how an inspection policy can be optimized. However the model uses a single random variable to represent the measurement and does not consider sensor measurement error. In the inspection process a sensor can contribute measurement error in addition to and independent of the natural variation in the container attribute values. The measurement error associated with inspection devices has a significant effect on the inspection decisions, and taking this into account would improve the model’s accuracy. Approaches to the treatment of measurement error in inspection are addressed in Chapter 2. We consider realistic situations where measurement errors exist and are embedded in the measurement readings obtained by the inspection devices. When a simple accept or reject threshold is used, containers with measurements close to the threshold value are at risk for misclassification. In this chapter we suggest the use of repeat inspections as a valid approach to reducing the impact of measurement errors on the performance of the inspection system.

This chapter is organized as follows. In Section 5.1, an analytical solution is illustrated for the cost minimization of an inspection model with no measurement error term. This is contrasted with the model presented in Section 5.2 which includes an independent
measurement error term. Examples of the updated model’s performance are given in Sections 5.2.1 and 5.2.2. A multi-objective problem is formulated in Section 5.3 and various solution methods are discussed. Finally Section 5.4 offers a discussion of the work presented.

5.1 Analytical Solution of Basic Inspection Problem

The fundamental inspection problem is to assess available information and evaluate the status of a container. The very nature of this process involves two inherent risks that are very often in direct conflict: accepting an undesirable container and subjecting an acceptable container to costly manual inspection, which we refer to as ‘rejection’ because the container is regarded as suspicious and leaves the general inspection process. We refer to the probabilities of these errors as PFA and PFR respectively.

Given a randomly selected container attribute of unknown status $x$, there is a probability $P(x = 1) = \pi$ that it contains undesirable cargo and a certain probability that the container is acceptable $P(x = 0) = 1 - \pi$ with respect to that attribute. Here $\pi$ can be estimated from information about the general population from which the container is selected for inspection. Similar to previous work, the distribution of the measured value of the container attribute $y$ is dependent on $x$. Let $f_0(y)$ represent the probability density function of measurement values $y$ given $x = 0$, and let $f_1(y)$ represent the pdf of $y| x = 1$. 
In a simple inspection procedure, the measurement is compared against a threshold value $T$ and containers with measurements falling below are deemed acceptable (not containing contraband), while containers with measurements above $T$ are deemed suspicious and subjected to manual inspection. Thus the probability of false acceptance ($PFA$) can be written as $PFA = P(y < T \mid x = 1)$ and the probability of false rejection ($PFR$) can be written as $PFR = P(y > T \mid x = 0)$. If we define the cumulative density functions $F_0(t) = \int_{-\infty}^{t} f_0(y)dy$ and $F_1(t) = \int_{-\infty}^{t} f_1(y)dy$, then the probabilities can be written $PFA = F_1(T)$ and $PFR = 1 - F_0(T)$. So as the threshold value increases $PFA$ increases and $PFR$ decreases and the reverse is true when the threshold value decreases.

To find an acceptable compromise we consider the effect from each type of misclassification and proceed to balance the effects. Cost is an obvious criterion, so we define $C_{FA}$ and $C_{FR}$ as the cost of false acceptance and false rejection (cost of additional scrutiny), respectively. The total expected cost can be calculated by considering the expected cost from both types of misclassification:

$$E[\text{Cost}] = \pi \cdot C_{FA} \cdot PFA + (1 - \pi) \cdot C_{FR} \cdot PFR$$

or

$$E[\text{Cost}] = \pi \cdot C_{FA} \cdot F_1(T) + (1 - \pi) \cdot C_{FR} \cdot (1 - F_0(T)) . \quad (5.1)$$

The minimum of Eq. (5.1) can be found by taking the derivative with respect to $T$ and setting it equal to zero:
\[
\frac{d}{dT}\left(\pi \cdot C_{FA} \cdot F_1(T) + (1- \pi) \cdot C_{FR} \cdot (1- F_0(T))\right) = 0
\]
\[
\pi \cdot C_{FA} \frac{d}{dT} F_1(T) + (1- \pi) \cdot C_{FR} \frac{d}{dT} (1- F_0(T)) = 0
\]
\[
\pi \cdot C_{FA} \cdot f_1(T) + (1- \pi) \cdot C_{FR} \cdot (-f_0(T)) = 0
\]
\[
\pi \cdot C_{FA} \cdot f_1(T) = (1- \pi) \cdot C_{FR} \cdot (f_0(T)).
\]

If we define the composite weight \( w_c = \frac{\pi \cdot C_{FA}}{(1- \pi) \cdot C_{FR}} \), the optimal threshold to achieve minimum expected cost is \( T^* : w_c f_1(T^*) = f_0(T^*) \). In other words, the optimum threshold value is such that the weighted pdf’s are equal. A closed form expression for the optimal value of \( T^* \) is difficult to obtain for normally distributed measurements, which we have assumed in previous research. Therefore we approximate the normal distribution with a Weibull distribution in order to solve the optimization analytically and investigate the effect of \( T \) on the \( PFA \) and \( PFR \). Note that the Weibull can be used to approximate the normal distribution when the shape parameter is about 3.35.

**5.1.1 Weibull Distributed Measurements**

In this section we determine the optimal threshold value for the simple inspection process described above using the following distributions:

\[
f_0(y \mid x = 0) = \frac{k}{\lambda_0} \left(\frac{y}{\lambda_0}\right)^{k-1} \exp\left(-\frac{y}{\lambda_0}\right),
\]

\[
f_1(y \mid x = 1) = \frac{k}{\lambda_1} \left(\frac{y}{\lambda_1}\right)^{k-1} \exp\left(-\frac{y}{\lambda_1}\right)
\]
for \( y \geq 0 \), where \( \lambda_0 \) is the scale parameter for measurements from acceptable containers, \( \lambda_1 \) is the scale parameter for measurements from acceptable containers, and \( k \) is the common shape parameter (since both distributions are approximating a normal, we assume the shape parameter is similar). Then the misclassification costs can be calculated as follows:

\[
PFA = F_1(T) = \int_0^T f_1(y) dy = 1 - \exp\left(-\left(\frac{T}{\lambda_1}\right)^k\right), \tag{5.5}
\]

\[
PFR = 1 - F_0(T) = 1 - \int_0^T f_0(y) dy = \exp\left(-\left(\frac{T}{\lambda_0}\right)^k\right). \tag{5.6}
\]

The expected cost is written as

\[
E[\text{Cost}] = \pi C_{FA} PFA + (1 - \pi) C_{FR} PFR
\]

\[
E[\text{Cost}] = \pi C_{FA} \left[1 - \exp\left(-\left(\frac{T}{\lambda_1}\right)^k\right)\right] + (1 - \pi) C_{FR} \exp\left(-\left(\frac{T}{\lambda_0}\right)^k\right). \tag{5.7}
\]

Taking the derivative of Eq. (5.7) with respect to \( T \) and setting equal to zero yields

\[
\pi C_{FA} \frac{k}{\lambda_1} \left(\frac{T}{\lambda_1}\right)^{k-1} \exp\left(-\left(\frac{T}{\lambda_1}\right)^k\right) + (1 - \pi) C_{FR} \left(-\frac{k}{\lambda_0}\right) \left(\frac{T}{\lambda_0}\right)^{k-1} \exp\left(-\left(\frac{T}{\lambda_0}\right)^k\right) \equiv 0. \tag{5.8}
\]

Eq. (5.8) can be solved to find the value for \( T \) that results in the minimum expected cost:
This solution can be used to approximate the optimal threshold in a problem where the measurement values are approximately normally distributed. For example, the following assumptions are made: \( \frac{\pi C_{FA}}{(1-\pi)C_{FR}} = 0.2 \), the distribution of \( y \mid x = 0 \) follows a Weibull distribution with shape parameter \( k = 3.439 \) and scale parameter \( \lambda_0 = 0.5 \), and \( y \mid x = 1 \) follows a Weibull with \( k = 3.439 \) and \( \lambda_1 = 1.0 \). Eq. (5.9) can be solved for the optimal threshold, yielding \( T_{opt} = 0.769 \). This data can be approximated with normal distribution parameters \( y \mid x = 0 \sim N(0.4495, 0.1445^2) \) and \( y \mid x = 1 \sim N(0.8989, 0.289^2) \). The true optimal value for the normal case is numerically found and is between \( T = 0.766 \) and \( T = 0.767 \).
5.2 Incorporation of Measurement Error

In this section we incorporate measurement error in the inspection model. A numerical example is conducted to illustrate the effect of measurement error on the performance of a single inspection station. Varying levels of error are simulated in a simple threshold comparison inspection in 5.2.1 and the results are shown in Fig. 5.1. The error term in the model is independent from the true measurement. The following notations are used in defining the problem of inspection with measurement error.

\[ x = \text{true attribute state} \quad \epsilon = \text{measurement error} \]
\[ y = \text{true attribute value} \mid x \quad r = \text{measurement returned by sensor} \mid x \]

The independence between the measurement error and the true attribute value has a significant impact which can be taken advantage of in the inspection approach. Consider an initial inspection, modeled by \( r = y + \epsilon \). For any subsequent inspections of the same attribute, the value of \( y \) does not change but the error value \( \epsilon \) would be replaced with a new \( \epsilon' \) related to the subsequent inspection, where \( \epsilon \) and \( \epsilon' \) are independent. Therefore the result of the subsequent inspection is modeled by \( r' = y + \epsilon' \). If the impact of the measurement error can be decreased, a more accurate observation of the true attribute value can be obtained. There are several methods by which this can be achieved.

When a simple pass/fail threshold is used as discussed in the previous work, containers with measurements close to the threshold value are at risk for misclassification. A
second inspection of all containers would reduce the impact of the measurement errors on misclassification but would be inefficient and unnecessary. However, applying a repeat inspection to containers which are at higher risk of misclassification could be a practical option. The areas under the pdf adjacent to the threshold value contribute the most to the misclassification probabilities, and these can be captured by defining an area \[
\left( T - \frac{b}{2}, T + \frac{b}{2} \right)
\] of width \( b \) around the threshold. In Zhu et al. (2010), \( b \) is called the “re-inspection band” since it is used to select containers for re-inspection. If the sensor measurements outside this band are classified as usual, the contribution to misclassification errors from the first inspection would be:

\[
PFA_1 = P\left( r < T - \frac{b}{2} \mid x = 1 \right) = P\left( y + \epsilon < T - \frac{b}{2} \mid x = 1 \right),
\]

\[
PFR_1 = P\left( r > T + \frac{b}{2} \mid x = 0 \right) = P\left( y + \epsilon > T + \frac{b}{2} \mid x = 0 \right).
\]

A second inspection is taken of the selected containers and the results are compared against \( T \) to make a final accept or reject decision. The misclassification errors resulting from the second inspection are:

\[
PFA_2 = P\left( r' < T, T - \frac{b}{2} \leq r \leq T + \frac{b}{2} \mid x = 1 \right)
= P\left( y + \epsilon' < T, T - \frac{b}{2} \leq y + \epsilon \leq T + \frac{b}{2} \mid x = 1 \right),
\]

\[
PFR_2 = P\left( r' > T, T - \frac{b}{2} \leq r \leq T + \frac{b}{2} \mid x = 0 \right)
= P\left( y + \epsilon' > T, T - \frac{b}{2} \leq y + \epsilon \leq T + \frac{b}{2} \mid x = 0 \right).
\]

Note that the probabilities in the second step contain conditions that are associated by the \( y \) term. This dependence makes it difficult to formulate and solve the optimization model
of this problem. The formulation involves the convolution of the density functions of $y$ and $\varepsilon$ and also of $y$ and $\varepsilon'$. Different approaches may include estimation, finding distributions that are amenable to the required calculations, and other methods for the analytic formulation of the expressions for $PFA_2$ and $PFR_2$.

However, if distributions for the terms are assumed, the problem can be formulated and solved using a program in MATLAB. For example, assume the following distributions:

$$y \mid x = 0 \sim N\left(0, \sigma_0^2\right), \quad y \mid x = 1 \sim N\left(1, \sigma_1^2\right), \quad \varepsilon \sim N\left(0, \sigma_e^2\right)$$

for the first inspection, and

$$\varepsilon' \sim N\left(0, \sigma_e'^2\right)$$

for the second. Since the error and attribute value are independent,

$$r \mid x = 0 \sim N\left(0, \sigma_0^2 + \sigma_e^2\right) \quad \text{and} \quad r \mid x = 1 \sim N\left(1, \sigma_1^2 + \sigma_e^2\right).$$

The misclassification probabilities can now be written in terms of the distribution parameters as follows:

$$PFA_1 + PFA_2 = \Phi\left(\frac{T - \frac{b}{2} - 1}{\sqrt{\sigma_1^2 + \sigma_e^2}}\right) +$$

$$\int \left[ \Phi\left(\frac{T - y}{\sigma_e'}\right) \left( \Phi\left(\frac{T + \frac{b}{2} - y}{\sigma_e}\right) - \Phi\left(\frac{T - \frac{b}{2} - y}{\sigma_e}\right) \right) \right] \frac{\phi\left(\frac{y - 1}{\sigma_1}\right)}{\sigma_1} dy,$$

(5.14)

$$PFR_1 + PFR_2 = 1 - \Phi\left(\frac{T + \frac{b}{2}}{\sqrt{\sigma_0^2 + \sigma_e^2}}\right) +$$

$$\int \left(1 - \Phi\left(\frac{T - y}{\sigma_e'}\right)\right) \left[ \Phi\left(\frac{T + \frac{b}{2} - y}{\sigma_e}\right) - \Phi\left(\frac{T - \frac{b}{2} - y}{\sigma_e}\right) \right] \frac{\phi\left(\frac{y}{\sigma_0}\right)}{\sigma_0} dy.$$

(5.15)

The effect of varying the threshold level $T$ and re-inspection band width $b$ on $PFA$ and $PFR$ is illustrated in a numerical example in Section 5.2.2.
5.2.1 Numerical Example with Varying Measurement Error

A sensor measurement, modeled by \( r = y + \varepsilon \), is compared against a preset threshold value \( T \) to make an accept (\( r < T \)) or reject (\( r > T \)) decision. The percent of unacceptable containers is \( P(x=1) = \pi = 0.00001 \). The following distributions are assumed: \( y \mid x = 0 \sim N(0,0.1^2) \), \( y \mid x = 1 \sim N(1,0.15^2) \), and \( \varepsilon \sim N(0,\sigma^2_{\varepsilon}) \). Threshold values in the range \([0,1]\) are simulated and represented by the horizontal axis. The series are composed of output resulting from different values of \( \sigma^2_{\varepsilon} \) ranging from 0.005 to 0.2. The two output values for each input combination of threshold and error are \( PFA = P(\text{accept} \mid x = 1) \), represented by the solid lines, and \( PFR = P(\text{reject} \mid x = 0) \), represented by the dashed lines as shown in Figure 5.1.
5.2.2 Numerical Example with Varying Re-Inspection Band

A two-step repeat inspection policy is defined that uses the parameters $T$ and $b$. In the first step an initial measurement is modeled by $r = y + \varepsilon$ and one of three decisions is made: accept when $r < T - \frac{b}{2}$, reject when $r > T + \frac{b}{2}$, and re-inspect using the same sensor when $T - \frac{b}{2} \leq r \leq T + \frac{b}{2}$. In the second step the measurement is modeled by $r' = y + \varepsilon'$ and an accept ($r' < T$) or reject ($r' > T$) decision is made. This example
uses the same assumptions as the previous example with the following exceptions:
\[ \varepsilon \sim N\left(0, 0.2^2\right) \] and \[ \varepsilon' \sim N\left(0, 0.005^2\right) \). Fig. 5.2 illustrates the effect of different \( b \) values (ranging from 0 to 0.5) and \( T \) on \( PFA \) values (solid lines) and \( PFR \) values (dashed lines).

![Figure 5.2. Performance of Repeat Inspection with Varying Re-inspection Band](image)

The inspection policies in the two numerical examples can be compared by observing performance data with similar settings. Table 5.1 compares the results for \( T = 0.45 \) in two particular cases for which the only difference between the policies is the extra step of a second more precise inspection.
Table 5.1. Inspection Policy Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>$PFA$</th>
<th>$PFR$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple inspection</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All attributes with $\sigma_\varepsilon = 0.2$</td>
<td>0.0139</td>
<td>0.0221</td>
</tr>
<tr>
<td><strong>Repeat inspection with $b = 0.1$</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All attributes with $\sigma_\varepsilon = 0.2$ and $\sigma_{\varepsilon'} = 0.005$</td>
<td>0.0082</td>
<td>0.0127</td>
</tr>
</tbody>
</table>

The results in Table 5.1 demonstrate the benefit (decrease in misclassifications) from the extra step of inspection, while the cost from the additional inspection increases with the percent of containers subjected to repeat inspection, which is 4.34% in this example. The policy defined for this example is capable of decreasing the impact of measurement error on the performance of the inspection system. Several other inspection policies involving repeat inspections may be defined and their performances compared in future research.

### 5.3 Multi-Objective Problem with Measurement Error

Again considering an inspection policy at a single station that includes a repeat inspection for containers with initial measurements falling within the band $b$, a multi-objective optimization problem can be formulated. Three important factors in this type of inspection policy are the $PFA$, $PFR$, and the percent of containers that are subject to re-inspection. The decision variables in this case are $b$ and $T$. The formulation is given as:
\[
\begin{align*}
\min z &= (f^{(1)}(b, T), f^{(2)}(b, T), f^{(3)}(b, T)) \\
\text{s.t.} \quad 0 &\leq T \leq 1 \\
&\quad 0 \leq b \leq 1,
\end{align*}
\]

where \( f^{(1)}(b, T) \) is the PFA, \( f^{(2)}(b, T) \) is the PFR, and \( f^{(3)}(b, T) \) is the percent of re-inspected containers. Eq. (5.16) minimizes the three objectives subject to constraints on the decision variables \( b \) and \( T \) in Eqs. (5.17) and (5.18).

We further express these objectives as:

\[
\begin{align*}
f^{(1)}(b, T) &= P\left(y + \varepsilon < T - \frac{b}{2} \mid x = 1\right) \\
&\quad + P\left(y + \varepsilon' < T, T - \frac{b}{2} \leq y + \varepsilon \leq T + \frac{b}{2} \mid x = 1\right) \\
f^{(2)}(b, T) &= P\left(y + \varepsilon > T + \frac{b}{2} \mid x = 0\right) \\
&\quad + P\left(y + \varepsilon' > T, T - \frac{b}{2} \leq y + \varepsilon \leq T + \frac{b}{2} \mid x = 0\right) \\
f^{(3)}(b, T) &= \pi P\left(T - \frac{b}{2} \leq y + \varepsilon \leq T + \frac{b}{2} \mid x = 1\right) \\
&\quad + (1 - \pi) P\left(T - \frac{b}{2} \leq y + \varepsilon \leq T + \frac{b}{2} \mid x = 0\right).
\end{align*}
\]

If the cumulative distribution functions of \( y + \varepsilon \mid x = 0 \) and \( y + \varepsilon \mid x = 1 \) are known, the percent of containers subject to re-inspection can be written more simply as:

\[
\begin{align*}
f^{(3)}(b, T) &= \pi \left(F_{y+\varepsilon\mid x=1}\left(T + \frac{b}{2}\right) - F_{y+\varepsilon\mid x=1}\left(T - \frac{b}{2}\right)\right) \\
&\quad + (1 - \pi) \left(F_{y+\varepsilon\mid x=0}\left(T + \frac{b}{2}\right) - F_{y+\varepsilon\mid x=0}\left(T - \frac{b}{2}\right)\right).
\end{align*}
\]
This multi-objective optimization problem can be solved using several methods. Among these is one that requires the assignment of cost values to each objective and solving the weighted sum objective. However, these cost values can be difficult to define and it is useful to be able to generate true multi-objective solutions, or Pareto optimal solutions. See the publications cited in Chapter 4 (Eschenauer et al. 1990, Statnikov and Matusov 1995, Fonseca and Fleming 1998a, 1998b, and Leung and Wang 2000) for the general treatment of multi-objective optimization problems.

Masin and Bukchin (2008) note that a common approach to the problem is to generate the efficient frontier by solving a weighted sum of the objectives for many different weight values, producing a set of Pareto optimal solutions, which is exactly what we did in Chapter 4. Two drawbacks to this method, as noted by (Masin and Bukchin 2008), are difficulty in generating the efficient frontier, and yielding too many points to handle in a subsequent decision-making process. Masin and Bukchin develop a “diversity maximization approach” (DMA) for multi-objective optimization. The goal in each iteration is to find the solution with a maximum diversity measure with respect to the current set of optimal solutions that form the efficient frontier. In the case of continuous input as we have, the DMA can be applied to produce an $\varepsilon$-approximation of the efficient frontier. The steps for adapting the DMA to solve the specific problem in Eq. (5.16) are outlined here. We use the following notation: $z = f(b,T)$, where the value of the $i^{th}$ objective of solution $(b,T)$ is $z^{(i)} = f^{(i)}(b,T)$ and we choose some $\varepsilon \geq 0$. Assume the
constraints on the values $b$ and $T$ apply throughout. Executing the following sequence will generate an $\varepsilon$-approximation of the efficient frontier:

**Step 1.** Solve

$$\min_Y \sum_{i=1}^{3} f^{(i)}(b, T)$$

s.t. $0 \leq b \leq 1$ and $0 \leq T \leq 1$

and let $z^* = f(b^*, T^*)$ be the optimal values. Set $E = \{z^*\}$.

**Step 2.** Solve

$$\min_Y \text{lex min} \left( \alpha, \sum_{i=1}^{3} f^{(i)}(b, T) \right)$$

s.t. $\alpha = \max_{z_e \in E} \left( \min_{i=1, 2, 3} \frac{f^{(i)}(b, T) - z_e^{(i)}}{z_e^{(i)}} \right)$

and let $z^* = f(b^*, T^*)$ be the optimal values.

**Step 3.** If $\alpha(z^*) < -\varepsilon$, then $E = E \cup z^*$, go to Step 2; else STOP.

Step 1 is used to find the initial point of the frontier; we applied equal weights for this step. In each iteration the absolute value of $\alpha(z^*)$ is less than or equal to its value in the previous iteration and the point $z^*$ is added to the partial efficient frontier $E$ until the condition in Step 3 is met and the iterations cease. See Masin and Bukchin (2008) for further details. The DMA could be further modified for use with this problem and it might be possible to develop a more efficient method relevant to our specific formulation.
5.4 Discussion

Measurement error has a significant effect on the decisions made following inspection and it is important to account for it. In this chapter we introduce an independent measurement error term in an updated model and investigate the impact of such errors on the system performance. We first demonstrate that an analytical solution to cost minimization for a model with no measurement error can be found. However if the enhanced model with measurement error term is applied to multiple inspections where selection for a second inspection is based on the results of the first, the probabilities from each step are not independent and no closed form solution is presented. Instead a numerical example simulates sensor measurements with varying levels of error to illustrate the effects on the performance of a single inspection station across several threshold levels. We then propose an inspection procedure that involves a re-inspection band and a second step in which the same inspection is repeated. The effect of this proposed repeat inspection procedure on the station performance is analyzed with another numerical example. The results of the example, with varying threshold and re-inspection band parameters, demonstrate the benefit in decreased misclassifications from the extra step of inspection and suggest that the repeat inspection procedure is a valid approach to reducing the impact of measurement errors on the performance of the inspection system. Finally we define three objectives of a repeat inspection policy and discuss novel solution methods for this multi-objective problem.
There are a few interesting directions for future work related to this measurement error model. Several other inspection policies involving repeat inspections can be defined and their performances may be compared. For example, Zhu et al. (2010) propose a few different policies and conduct experiments to compare their performances. As stated, one could modify the DMA discussed in Section 5.3 to develop a more efficient method relevant to that specific formulation.
CHAPTER 6

OPTIMAL ALLOCATION AND SCHEDULING OF INSPECTION OPERATIONS UNDER MULTIPLE RISK CATEGORIES

In this chapter we consider the situation where unacceptable container cargo is classified into \( k \) risk categories, for example: biological, radiological, chemical, and nuclear. Containers are profiled based on many factors such as origin, destination, shipper fidelity, and intelligence from various sources. This information is used to generate the risk profile for a given container, in which each component estimates the perceived likelihood of a container including unacceptable cargo of a certain type. This measure is interpreted as a probability for each risk category and together they form the ‘risk profile’ for a container. The allocation of inspection operations for each container can then be made accordingly. Optimization of inspection allocation given risk profiles of multiple risk categories may be carried out with inspection system constraints such as capacities and maximum false alarm probabilities. However, another important constraint on the inspection system is the timeframe in which containers are available for inspection according to their ‘due times.’ These represent the deadline for a container to be released, either for pickup by the receiving company or transport to a vessel for travel to the next destination. The expedition of container transportation and delivery has a great impact on the global supply chain network, making it critical to meet the imposed due
times. Not meeting due times may result in financial loss for the receiving company, penalties to the port operator, or overtime costs for the inspection provider.

In this chapter we consider the problem where containers have specified due times and penalties are incurred when these times are exceeded. We introduce a new formulation that corresponds to the consideration of multiple, independent risk categories in container inspection and investigate and develop algorithms that determine the optimal inspection allocation and schedule that minimizes the penalty due to container delays and the probability of false acceptance.

We begin by assuming a given allocation of containers to inspection stations and finding the optimal schedule to minimize tardiness. This problem resembles the well known open shop scheduling problem (OSSP) which is discussed in Section 6.2. Next, the formulation is presented for the problem where the allocation of containers is undefined. This formulation is found to be non-convex and the solution NP-hard so we describe our approach to the problem. This involves using complete enumeration to solve small size problems (in terms of the number of inspection stations and the number of containers) that serve as a base for comparison with heuristics for optimal and near-optimal solutions that make use of the problem’s relationship to OSSPs. This is first done for a special case of the problem with equal processing times; the problem is then expanded to consider unequal inspection times, which may be related to container risk profiles. This represents a much more challenging OSSP for which no polynomial time solution algorithms are known. New heuristics are created and the results are compared with the optimal
solutions obtained from a complete enumeration algorithm. Numerical examples are provided to test the heuristic performances for many different combinations of problem parameters.

6.1 Multiple Risk Categories

The concept of multiple risks changes the nature of the inspection problem and its formulation. Whereas previously $x = 0,1$ is used to indicate the true status of a container being acceptable or unacceptable, respectively, we now consider $x = (x_1, x_2, ..., x_k)$ as a vector of the true statuses with respect to each of the $k$ risk categories. Each of these categories represents a different type of risk with associated costs, preferences, etc. A container that has unacceptable cargo in any category $j$ (reflected by the status $x_j = 1$) is considered unacceptable overall. The probability of each risk category being present in a container is given by the vector $\pi = (\pi_1, \pi_2, ..., \pi_k)$ where $\pi_j = P(x_j = 1), j = 1, 2, ..., k$. These probabilities are assumed to be independent. Although there could be positive $\pi_j$ values for multiple $j$, it is assumed that no container would actually contain cargo from more than one risk category. The probability of such a scenario occurring is extremely small, however if the container is subject to manual inspection for any reason then all types of unacceptable cargo would be found. We refer to $\pi$ as a risk profile and assume that each container $i$ has an associated $\pi_i$. Because more information is available for individual containers, a tailored container inspection policy may be established for each container. In this problem, a number or pattern of inspections required to be completed is not
predefined, which differs from our earlier work. Instead of applying a predetermined Boolean system decision function as we have in previous chapters, we assume that a container that fails an individual inspection station then undergoes manual inspection, and a container that does not fail any stations is accepted for entry through the port.

The overall risk of a container with estimated \( \pi = (\pi_1, \pi_2, \ldots, \pi_k) \) being undesired and accepted is \( \sum_{j=1}^{k} \pi_j \) without any inspection. Assume that there are \( k \) inspection stations available and each specializes in the inspection of one type of the \( k \) risk categories, i.e. inspection station \( j \) is capable of detecting cargo associated with risk \( j \) only. The overall risk can be reduced by selecting inspection stations to analyze the container. Inspection is generally subject to errors and we define the probability of not detecting undesired cargo at station \( j \) to be \( PFA_j = P(\text{accept}_j | x_j = 1) = 1 - PTR_j \). Thus the actual information to be gained from inspection at station \( j \) is \( \pi_j PTR_j \), denoting the probability and detection of \( x_j = 1 \). If inspection resources were unlimited, containers could be inspected for all categories of risk to maximize the probability of detecting unacceptable containers. However, if some constraints are imposed or it is desired to balance system performance with cost, a subset of inspection stations must be selected for each container.

We define multiple risk categories and risk profiles from shipping data and intelligence information. Because containers have different allocations and inspection sequences and there is competition for time slots at inspection stations, all of the individual inspection
operations must be scheduled. This guarantees that they can be feasibly accomplished and defines the inspection completion time for each container.

6.2 Literature Review and Related Work

We first assume that the inspection allocation to be carried out on a batch of containers is defined and that each container has a specified due time. The scheduling of inspection may be done in order to minimize tardiness of containers, or other objectives. This problem can be considered an open shop scheduling problem, as defined by Kubiak et al. (1991) and Brucker (1995). Inspection stations are similar to machines in the classic OSSP, containers replace jobs, and each individual inspection operation is similar to a job operation. The classification scheme \( \alpha | \beta | \gamma \) is used to describe open shop scheduling problems. Assuming a given allocation, our inspection scheduling problem is represented as \( O3 | \beta | \sum T_i \), with \( \beta = \{ t_{ij} = 0, 1 \} \) for the first part of the work and \( \beta = \{ \} \) for the second. We assume non-preemptive operations, which limits the related work. Liu and Bulfin (1988), Kubiak et al. (1991), and Brucker et al. (1993) define the complexity of some OSSP problems and present solution methods for limited problem types. Different methods of representing and building schedules are presented in Brasel and Kleinau (1992) and Brasel et al. (2008). Brasel and Kleinau (1992) investigate the number of feasible schedules in OSSPs.

Following the same notations as in the scheduling literature, we assume that the due time \( d_i \) is given for container \( i \), at which time all inspection operations should be completed.
The completion time of container $i$ is denoted $C_i$ and the tardiness of container $i$ is defined as $T_i = \max(0, C_i - d_i)$. So, if inspection operations are completed after $d_i$, the tardiness of that container increases and results in an increased penalty in the objective function. Tardiness is a useful objective in situations where there is no impact (negative or positive) from completing operations before the expected due time, such as in the container inspection problem where there is no benefit or penalty that results from finishing inspection early.

It should be noted that minimum makespan is a widely used objective in the OSSP literature. We are assuming that in the container inspection problem, meeting container deadlines is a more relevant objective than reducing the time for completion of inspection for all containers, especially in cases where inspection time is allotted in advance and reducing the time actually used below this allotment would not impart any additional savings. Brucker and Knust (2009) provide a thorough review of various scheduling problems and from this it is clear that there are limited open shop problems with a tardiness objective for which efficient solution methods have been found—see Liu and Bulfin (1988) and Timkovsky (2003).

6.3 Problem Formulation

When the allocation of inspection operations is not known beforehand, the scheduling problem cannot be stated since the operations are not defined. Thus we return to the original problem of simultaneous inspection allocation and scheduling. The goal is to
determine an optimal allocation by selecting the inspection operations that have the
greatest effect on false acceptance probabilities, while weighing the effect on the optimal
schedule and tardiness.

We introduce the following notations to represent the problem mathematically:

- \(i\) container \(i\) \((i=1,2,...,n)\)
- \(j\) inspection station \(j\) \((j=1,2,...,k)\)
- \(\pi_{ij}\) probability of risk category \(j\) being present in container \(i\)
- \(PFR_j\) probability of false reject at station \(j\)
- \(PTR_j\) probability of true reject at station \(j\)
- \(y_{ij}\) = \[
\begin{cases} 
1 & \text{if container } i \text{ is inspected by station } j \\
0 & \text{otherwise}
\end{cases}
\]
- \(d_i\) due time of container \(i\)
- \(t_{ij}\) processing (inspection) time of container \(i\) at station \(j\)
- \(s_{ij}\) start time of inspection of container \(i\) at station \(j\)
- \(C_i\) completion time of container \(i\) inspection at all allocated stations
- \(T_i\) tardiness of container \(i\)
- \(M\) a large positive number
- \(w_T\) weight of tardiness in objective function
- \(z_{ijg}\) = \[
\begin{cases} 
1 & \text{if container } i \text{ precedes container } g \text{ on station } j \\
0 & \text{otherwise}
\end{cases}
\]
- \(zz_{jhi}\) = \[
\begin{cases} 
1 & \text{if station } j \text{ precedes station } h \text{ in inspection of container } i \\
0 & \text{otherwise}
\end{cases}
\]
The inspection system consists of \( k \) stations which may be used in the inspection of containers before they are accepted for clearance through a port. Station \( j \) corresponds to risk category \( j \) and is effective in detecting cargo of type \( j \). There are \( n \) containers and each one is associated with a risk profile. As defined in Section 6.1 a risk profile is the perceived likelihood of a container including unacceptable cargo of each of the \( k \) defined categories. Each profile is interpreted from intelligence and container information and provided as an array of probabilities. There is no predetermined allocation of containers to inspection stations. The formulation for allocation and scheduling of the containers that minimizes the sum of false acceptance probabilities (i.e., the probability that undesired cargo of type \( j \) is present and undetected in a container following completion of inspection) over all risk categories \( j = 1, 2, \ldots, k \) and weighted tardiness is given as follows:

\[
\min \left(1 - w_T\right) \sum_{i=1}^{n} \sum_{j=1}^{k} \pi_{ij} \left[1 - PTR_j y_{ij}\right] \prod_{s=1}^{k} \left[1 - PFR_{s, y_{is}}\right] + w_T \sum_{i=1}^{n} T_i \quad (6.1)
\]

s.t. \( s_{ij} + t_{ij} y_{ij} - d_i - T_i \leq 0 \quad \forall i = 1, 2, \ldots, n; \forall j = 1, 2, \ldots, k \quad (6.2) \)

\[
s_{ij} + t_{ij} y_{ij} - M \left(1 - z_{igj}\right) - M \left(1 - y_{ij}\right) - s_{gj} \leq 0 \quad \forall i \neq g, \forall j = 1, 2, \ldots, k \quad (6.3)
\]

\[
s_{ij} + t_{ij} y_{ij} - M \left(1 - z_{jih}\right) - M \left(1 - y_{ih}\right) - s_{ih} \leq 0 \quad \forall i = 1, 2, \ldots, n, \forall j \neq h \quad (6.4)
\]

where:

\( y_{ij}, z_{igj}, z_{jih} = 0 \) or \( 1 \)

\( s_i, T_i \geq 0 \)

The first part of the objective function in Eq. (6.1) represents the penalty for accepting an undesired container while the second part represents the total tardiness. Constraint (6.2)
ensures that the completion time of any container does not exceed the due time plus
tardiness of that container. Constraint (6.3) ensures that a station does not inspect two
containers simultaneously. Similarly constraint (6.4) ensures that a container is not
inspected by any two stations at the same time. Eqs. (6.3) and (6.4) generate a large
number of constraints to consider all possible sequences of stations processing containers
and containers visiting stations. For example, if containers $i$ and $g$ are both inspected by
station $j \left(y_{ij} = y_{gj} = 1\right)$, the following constraints should be introduced to guarantee that
both containers are not processed simultaneously on $j$.

If $i$ precedes $g$ on station $j$, the following constraint must be met:

$$s_{ij} + t_{ij} \leq s_{gj}.$$  

And if $g$ precedes $i$ on station $j$, the following constraint must be met:

$$s_{gj} + t_{gj} \leq s_{ij}.$$  

Similar disjunctive constraints are generated if container $i$ is inspected by two stations $j$
and $h$, when $y_{ij} = y_{ih} = 1$.

If $i$ is inspected on station $j$ before station $h$, the following constraint must be met:

$$s_{ij} + t_{ij} \leq s_{ih}.$$  

And if $i$ visits station $h$ before station $j$, the following constraint must be met:

$$s_{ih} + t_{ih} \leq s_{ij}.$$  

To accommodate the disjunctive nature of these required constraints the terms
$M \left(1 - z_{ij} \right)$ and $M \left(1 - zz_{jih} \right)$ are included in Eqs. (6.3) and (6.4), respectively. Also, the
constraints in the above example are only valid when both containers are inspected on the same station, or \( y_{ij} = y_{gj} = 1 \). In the case of one container being allocated for inspection by two stations, \( y_{ij} = y_{ih} = 1 \) must be true. To reflect these conditions the terms \( M(1 - y_{ij}) \) and \( M(1 - y_{ih}) \) are included in Eqs. (6.3) and (6.4), respectively, and the term \( t_{ij}y_{ij} \) appears in both equations.

The binary values \( y_{ij} \) defined for \( i = 1, 2, \ldots n \) and \( j = 1, 2, \ldots k \) are decision variables that collectively define an inspection allocation dictating which containers are inspected by which stations, while the start times \( s_{ij} \) outline the scheduling of these operations. It should be noted that \( s_{ij}, z_{igj}, \) and \( z_{jhi} \) are decision variables that do not appear in the objective function. These variables affect the tardiness \( T_i \) of each container, so they are implicitly included in the objective.

### 6.4 Approach

Determining the optimal allocation and schedule simultaneously is a non-convex problem, therefore finding an optimal solution is difficult if not impossible when the number of inspection stations or containers is large. We investigate heuristics for optimal and near-optimal solutions that make use of the problem’s relationship to OSSPs. An important aspect of the heuristics is breaking the problem down to smaller, convex problems that can be solved and combined to obtain solutions to the original problem. This is achieved by considering the allocation of containers to inspection stations first,
then determining the best inspection schedule. Thus the inspection scheduling problem becomes an open shop scheduling problem as defined in Section 6.2 and approaches used in the OSSP literature can be applied to find an optimal schedule.

The allocation and scheduling problem stated in Section 6.3 is formulated as a mixed integer nonlinear program (MINLP) in the General Algebraic Modeling System (GAMS). However, the problem is non-convex and therefore the solver algorithms are heuristics and an optimal solution is not guaranteed. Several small size problems are solved with complete enumeration and these solutions are compared to solutions from a proposed heuristic to evaluate the performance of the heuristic. Calculating a theoretical lower bound for the tardiness resulting from a given allocation is important for gauging the optimality of that allocation and for limiting the time spent searching for optimal schedules. Also, minimum and maximum values for the two objectives of \( \sum PFA \) and tardiness must be found and normalized so that meaningful values of \( w_T \) can be chosen. The following sections detail the lower bound calculations and normalization method as well as the proposed greedy search heuristic and complete enumeration algorithms.

6.4.1 Lower Bound on Tardiness Objective

Given an inspection allocation \( y \), we define a theoretical lower bound on the tardiness for an optimal schedule of \( y \). Application of this lower bound within an algorithm allows partial solutions to the full problem to be compared for optimality, thereby increasing algorithm efficiency. That is, allocations can be compared without first finding a feasible
schedule. When a close approximation of the theoretical lower bound is applied, the gain in efficiency is even greater. We consider two lower bounds which are described in detail as follows.

The first lower bound $LB_1$ can be formulated by considering individual containers. Each container must complete all inspections called for by allocation $y$, therefore the minimum completion time $C_i \geq \sum_{j=1}^{k} y_{ij} t_{ij}$. If the total time of inspections for a container exceeds its due time, the container experiences some positive tardiness. The summation of minimum tardiness across all containers provides a lower bound on the total tardiness for $y$, presented in Eq. (6.5).

$$LB_1 = \sum_{i=1}^{n} \max \left( 0, \sum_{j=1}^{k} y_{ij} t_{ij} - d_i \right)$$ (6.5)

The second lower bound $LB_2$ is formulated by considering inspection stations. The earliest time a station can complete all assigned inspections is $\sum_{i=1}^{n} y_{ij} t_{ij}$. If this time exceeds the latest due time of all containers inspected at that station, the station contributes some positive tardiness to the objective. In addition, tardiness can result from not meeting due times other than the latest one. For example, at some earlier due time $d_e$, consider the sum of assigned inspection times for containers with due time $d_i \leq d_e$. This is the time that station $j$ will complete inspection on the last container with $d_i = d_e$, and if this time exceeds $d_e$ then even under an optimal arrangement, some containers will exceed
this due time and contribute positive tardiness. To find the lower bound including these sources of tardiness an algorithm is used to find the minimum overdue time of containers, presented in Fig. 6.1. The overdue measure \( od_{jf} \) defined in the algorithm is the minimum tardiness for due time \( d_{jf} \) and the summation of this contribution from all unique due times of inspections at a station provides the minimum possible tardiness achievable from that station.

For each station \( j=1:k \)
- Order the unique due times of containers inspected at \( j \), from earliest \( \left( d_{[1]} \right) \) to latest \( \left( d_{[m_j]} \right) \)

For each unique due time \( d_{(f)} \), \( f=1:m_j \)
- Define the set of containers with that due time, \( S_{(f)} = \{ i : d_i = d_{(f)} \} \) and denote the size of \( S_{(f)} \) as \( \#S_{(f)} \)
- Define the set of containers with earlier due times \( R_{(f)} = \{ i : d_i < d_{(f)} \} \)
- Define an overdue measure
  \[
  od_{(f)} = \max \left( 0, \sum_{i \in S_{(f)} \cap R_{(f)}} y_{ij} t_{ij} - d_{(f)} \right)
  \]
  which is the minimum tardiness for due time \( d_{(f)} \)
- Define the number of overdue containers
  \[
  n_{(f)} = \min \left( od_{(f)}, \#S_{(f)} \right)
  \]

Figure 6.1. Lower Bound Algorithm

The algorithm used in determining \( LB_2 \) is presented in Fig. 6.1. This algorithm uses the allocation \( y \), inspection times \( t_{ij} \), and due time array. The assumptions used to arrive at
the theoretical lower bound on tardiness are that inspection operations can be carried out
at each station in order of earliest due time (EDT) and that there are no conflicts
(instances of a container being scheduled for inspection by multiple stations
simultaneously) that cannot be resolved. Under these assumptions we find the minimum
overflow when container inspection times exceed due times.

To continue formulating $LB_2$ we first consider the special case problem where the
processing (inspection) times are of unit length. A lower bound on the total system
tardiness is constructed from the summation of the overdue measure $od_{fj}$ from all unique
due times and across all stations; this lower bound for the special case is presented in Eq.
(6.6). Also, because $t_{fj} = 1$ in the special case, $LB_2$ can be tightened by considering limits
on the ability to overlap containers with positive tardiness across stations to minimize
$\sum T_i$. If $od_{fj}$ and $n_{fj}$ are greater than the number of stations $k$ for any $f$, $LB_2$ is increased by
$LB_2^+$ to reflect the fact that at most $k$ containers can be overlapped at the stations in a
range of $k$ time blocks and any additional containers will raise the theoretical minimum
tardiness. The amount by which $LB_2$ should be increased is presented in Eq. (6.7).

\[
LB_2 = \sum_{j=1}^{k} \sum_{f=1}^{m_j} od_{fj} \quad (6.6)
\]

\[
LB_2^+ = \sum_{j=1}^{k} \sum_{f=1}^{m_j} \sum_{g=1}^{\left[\frac{n_{fj}}{k}\right]} \max\left(od_{fj} - gk, 0\right) I\left(n_{fj} > gk\right) \quad (6.7)
\]

For the general case of problems without constraint on processing times the interaction
between containers at different stations is much more complicated and it becomes
difficult to define a precise lower bound. We develop a method in which the order of containers at each station is first considered independently, and assume there will be no conflicts across stations. The optimal order for scheduling operations at a single station is described in Brucker (1995) and we implement the sequence building algorithm to schedule all inspections at each station independent of the other stations. The tardiness from this best possible schedule is summed across all containers inspected at a station, resulting in the lowest achievable tardiness at each station, and the maximum of these $k$ values is taken as a lower bound for the general case, $LB_2 = \max_{1 \leq j \leq k} \sum_{i, \text{ inspected at } j} od_{ij}$. The overdue measure $od_{ij}$ used here results from the optimal order of container inspections and is defined for every container $i$ that is inspected at station $j$.

Since $LB_1$ and $LB_2$ are both theoretical lower bounds the greater of the two is used as an overall lower bound on the tardiness. Application of this lower bound improves the efficiency of every algorithm used in this chapter by restricting how often the associated scheduling subroutine is called.

6.4.2 Normalization of Objectives

The objective function consists of $\sum PFA$, which we refer to as risk, and the sum of individual container tardiness, $\sum T_i$, which we refer to as tardiness. The parameter $w_T$ represents the importance of tardiness vs. risk as a weight and is used to linearly combine the two objectives. Therefore it becomes important that the individual objectives of risk
and tardiness are normalized. Given problem parameters the minimum and maximum potential values of $R_{\text{min}}$, $T_{\text{min}}$, $R_{\text{max}}$, and $T_{\text{max}}$ are determined and used to translate risk and tardiness values to normalized values: $\frac{\text{Calculated} - \text{Min}}{\text{Max} - \text{Min}}$ that will fall in the range $(0,1)$. When the two objectives are normalized the meaning of $w_T$ is clear. The minimum and maximum values are established as functions of corresponding inspection allocations, and the method for calculating them is described as follows.

The upper limit of inspection is $y_{ij} = 1$ for all $i$ and $j$. The risk associated with this allocation is calculated and used as the minimum possible risk value, $R_{\text{min}}$. The tardiness resulting from a feasible schedule of this allocation would be the maximum tardiness of any allocation. The value for $T_{\text{max}}$ is found by generating a schedule that is feasible but not necessarily optimal.

To define the lower limit of inspection a basic scheduling procedure builds a feasible schedule of inspection operations, chosen in a general order by proceeding through the columns and rows of a potential allocation matrix. The schedule is created with no tardiness, therefore $T_{\text{min}} = 0$. The maximum risk $R_{\text{max}}$ is calculated from the allocation $y_{\text{low}}$ generated by the basic scheduling procedure. Because the inspection operations are chosen in a general way the risk objective of any potential optimal solution should meet or exceed the performance of $y_{\text{low}}$. 
6.4.3 Greedy Search Heuristic

A greedy search method selects from some prospective options the one that when added to the solution results in the best objective function. We apply a greedy search method to solve the allocation and scheduling problem by comparing prospective inspection operations and selecting a subset, thereby defining an inspection allocation. The algorithm begins with an empty allocation matrix and inspections are added one at a time until an optimal allocation is reached, at which point adding any further operations would increase the tardiness penalty more than the risk objective would decrease. Different heuristics are developed for special and general case problems, and scheduling procedures for each case are also created. The search and scheduling heuristics for the special and general cases are described in detail in the following paragraphs.

The greedy heuristic for the special case searches for the next inspection operation to add by ranking all prospective operations by a simple approximation of their maximum potential benefit to the objective function. A selection of operations with approximations falling within a defined range [maximum, maximum – epsilon] are identified as candidates. Each candidate is added one at a time to the current optimal allocation for evaluation, and subroutines calculate the risk and lower bound on tardiness. This is repeated for all candidate operations and the results are compared to find the best potential improvement in the objective function. A subroutine carries out the scheduling heuristic, described in the next paragraph, to determine the minimum tardiness that can be achieved in a feasible schedule. If one candidate is found to outperform the others in
decreasing the objective function, the optimal solution is updated with the new allocation, schedule, and objective function value. Operations that have been included in the optimal allocation are no longer considered as prospective operations. If none of the candidate operations in a search loop improve the current optimal, they are excluded from future searches. The algorithm ends when all prospective operations are excluded. Various epsilon values are implemented, resulting in multiple solutions and the best of these is returned as the overall solution of the special case greedy heuristic.

The scheduling heuristic designed for the special case takes advantage of the constraint of unit processing times by considering all inspection operations as equal time blocks. It proceeds through all operations in a given allocation and assigns each one a start time in a feasible schedule. The schedule is constructed to be feasible in that no conflicts exist across stations (inspecting more than one container at a given time) or containers (simultaneous inspection at multiple stations). The operations are chosen to be scheduled in such a way to ensure the final schedule has minimal deviation from EDD order and therefore minimizes total tardiness. The MATLAB functions that implement the greedy search and scheduling heuristics for special case problems can be found in Appendix D.

In the general case greedy heuristic the processing time of each prospective operation is an important factor that must be considered in both allocation and scheduling. In general, an operation with higher potential to reduce risk (using the same approximation of potential described above for the special case greedy heuristic) and lower processing time is preferred to one with lower potential and higher processing time. The greedy heuristic
for the general case considers a ratio of potential risk improvement over processing time and selects the operation with the highest ratio out of all available operations. This candidate operation is added to the current optimal allocation for evaluation and a scheduling heuristic (described in the next paragraph) is used to find the tardiness of an optimal schedule for that allocation. If the addition of the candidate operation results in a decreased objective function, the optimal solution is updated with the new allocation, schedule, and objective function value; if not the operation is excluded from future searches. Loops are implemented in which powers are applied to the numerator or denominator of the ratio to change the preference by which operations are selected. The allocation and schedule with the best objective function value over all loops is returned as the solution of the general case greedy heuristic.

The scheduling heuristic designed for the general case implements several loops that each construct a feasible schedule, and the schedule with minimum tardiness is chosen over all loops. Each loop uses a different method to specify the preferred sequence for containers to be inspected at each station. The methods include shortest processing time first, earliest due time first, most “busy” container first (where the busyness of a container $i$ is defined as the sum of processing times for operations across all stations, $\sum_{j=1}^{k} y_{ij} t_{ij}$), and assigning order by weighted combinations of these parameters. An additional method is applying the procedure from Brucker (1995) discussed in Section 6.4.1 to find the optimal sequence at each station independent of other stations. After the preferred sequence is specified, the heuristic assigns each operation a start time in a feasible
schedule, avoiding conflicts across stations and containers. Potential conflicts are avoided by delaying the assigned start time of a container to another conflict-free time and allowing another container to be scheduled ahead of it. The MATLAB functions that implement the greedy search and scheduling heuristics for general case problems can be found in Appendix E.

6.4.4 Complete Enumeration

The complete enumeration (CE) approach searches the entire solution space to guarantee an optimal solution to serve as a baseline for comparing the heuristic. The CE procedure starts by enumerating all possible values for a row of \( y \), which describes one container’s allocation for inspection at \( k \) stations. This is represented as a binary array of length \( k \), therefore there are \( 2^k \) potential row arrays. An allocation \( y \) consists of \( n \) rows, representing \( n \) containers, and for each one a potential row array is assigned. Subroutines are used to calculate the risk and lower bound of tardiness for \( y \). If the weighted objective function of these two measures is less than the current optimal objective, the scheduling algorithm is run as described in the next paragraph. Applying the lower bound on tardiness to assess the overall solution potential limits the frequency with which the scheduling subroutine is called, thereby reducing the running time of the program. A feasible schedule and tardiness is returned from the scheduler subroutine and if the objective function value including the tardiness is less than optimal objective found so far, the optimal solution is updated. The algorithm for the CE procedure contains \( n \) loops
to assign the rows of the allocation $y$ and each loop cycles through $2^k$ possible assignments to the $k$ stations.

The CE scheduling procedure is run at most once for every possible allocation $y$. The procedure enumerates all possible orders in which to perform the scheduling of the inspections allocated in $y$. For a given order, each inspection $y_{ij} = 1$ is assigned a position in the schedule at the earliest time that the container $i$ and station $j$ are available (not already assigned for inspection). A feasible schedule and the corresponding tardiness of that schedule are returned after all inspection operations are assigned. The CE scheduling procedure for special case problems is designed to take advantage of the unit processing times and considers all inspection operations as equal time blocks in the schedule. The MATLAB functions that carry out the CE search and scheduling procedures for special case problems can be found in Appendix F and the MATLAB functions for the general case CE search and scheduling procedures can be found in Appendix G.

6.5 Numerical Examples

Several numerical examples are generated and solved with the greedy heuristic and complete enumeration methods to demonstrate the performance of each for problems in both the special and general cases. Experimental values of some parameters such as $n$ and $k$ are limited by the computation time required by the complete enumeration method. Random values for risk profiles are independently generated and uniformly distributed
between 1E-5 and 1E-2. The computation time and solution performance results are detailed in the following sections.

6.5.1 Computation Time

Several examples are run with the complete enumeration method to demonstrate the computation time of the program as a function of varying problem parameters. The first set of experiments is for a three station inspection system, with the number of containers varying between 5 and 10. A due code \((dc)\) of 4 indicates that the due times are equal to 4, and \(w_T\) is 0.99. We start with the special case problem where the processing (inspection) times are of unit length. Table 6.1 presents the computation time of the complete enumeration program running these experiments.

<table>
<thead>
<tr>
<th>(n)</th>
<th>Computation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>831</td>
</tr>
<tr>
<td>8</td>
<td>6635</td>
</tr>
<tr>
<td>9</td>
<td>53934</td>
</tr>
<tr>
<td>10</td>
<td>363453</td>
</tr>
</tbody>
</table>

The computation times increase roughly by a factor of 8 for each additional container, which corresponds to an additional loop in the CE program which has \(2^3\) possible choices for allocation. It is fairly obvious from this example that the CE program quickly
becomes intractable with larger parameter values. Similar results are obtained for experiments with different due times. There is a general pattern observed of slightly increasing program computation time with larger due times. Another set of experiments is run for four stations for the special case, with due times equal to 4 and $w_T = 0.99$. Table 6.2 presents the computation time of these experiments.

Table 6.2. Computation Time of CE for Special Case with $k = 4, dc = 4$

<table>
<thead>
<tr>
<th>$n$</th>
<th>Computation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>95</td>
</tr>
<tr>
<td>6</td>
<td>1544</td>
</tr>
<tr>
<td>7</td>
<td>25711</td>
</tr>
</tbody>
</table>

The computation times in Table 6.2 increase roughly by a factor of 16 for each additional container. This corresponds to the addition of a loop in the CE program with $2^4$ possible allocation choices for a new container. Similar to the first experiment, there is a slight increase in computation times with higher due times. From the results of the first two experiments with $w_T = 0.99$ it is observed that the computation time for the CE algorithm can be modeled by $c2^{nk}$ where $c$ is a constant, $n$ is the number of containers, and $k$ is the number of inspection stations. Because it is an exponential time algorithm the solution of larger problems quickly becomes intractable.

We eliminate the constraint of unit length processing time and consider the general case of problems by generating uniformly distributed integers for the processing times. The
complete enumeration scheduling procedure used for these problems is much more complex and the computation time is longer. To demonstrate the CE program computation time for the general case, experiments are carried out for 3 and 4 stations with the number of containers varying between 5 and 8. A due code of 7 indicates the due times are equal to 7, and $wT=0.99$. Table 6.3 presents the computation time in seconds of the CE program for the general case experiments.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k=3$</th>
<th>$k=4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>59</td>
<td>2259</td>
</tr>
<tr>
<td>6</td>
<td>632</td>
<td>52925</td>
</tr>
<tr>
<td>7</td>
<td>8656</td>
<td>1514509</td>
</tr>
<tr>
<td>8</td>
<td>112130</td>
<td>-</td>
</tr>
</tbody>
</table>

The times found in Table 6.3 are greater than the times for corresponding size problems given in Tables 6.1 and 6.2. There is a pattern of increasing computation time with increasing $n$ and $k$ in Table 6.3, however it is not as clearly explicable as the pattern distinguished in the special case experiments. This is likely due to the fact that the scheduling procedure for the general case has more complex loops than the one designed for the special case.

When a smaller $wT$ value is used, the computation time increases because more allocations are viable competitors to improve the objective function during the CE search, which leads to more frequent use of the enumerative scheduling algorithm, thereby
increasing the time required to obtain the optimal solution. Some example computation times for the general case are given in Table 6.4 for experiments of five containers in a three station system with due times equal to 6 and varying \( wT \) values.

<table>
<thead>
<tr>
<th>( wT )</th>
<th>Computation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.99</td>
<td>59</td>
</tr>
<tr>
<td>0.60</td>
<td>993</td>
</tr>
<tr>
<td>0.40</td>
<td>2273</td>
</tr>
</tbody>
</table>

Table 6.4. Computation Time of CE for General Case under Various \( wT \) Values

As \( w_T \) decreases, more allocations appear competitive and more scheduling loops are run, resulting in the increased pattern of computation time displayed in Table 6.4. The pattern of increase in computation time is difficult to quantify because it depends on the order in which allocations are tested and when lower values for the objective function are found, which limits the time spent considering future potential solutions. While it is difficult to quantify the effect of lower \( w_T \) values on the computation time, it is certain that the time increases and will be some multiple of exponential time with respect to the parameters \( n \) and \( k \).

The greedy heuristic is polynomial time because it constructs a feasible allocation one inspection at a time, so the number of overall loops is limited by the total number of possible inspections, \( nk \). The subroutines, including the scheduler algorithm, are also polynomial time. Computation times in seconds of the greedy heuristic (GH) program
for the special case are presented in Table 6.5 for systems of three and four stations, with parameters matching those of the experiments presented in Tables 6.1 and 6.2.

Table 6.5. Computation Time of GH for Special Case with $dc = 4$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.07</td>
<td>0.12</td>
</tr>
<tr>
<td>6</td>
<td>0.07</td>
<td>0.13</td>
</tr>
<tr>
<td>7</td>
<td>0.08</td>
<td>0.14</td>
</tr>
<tr>
<td>8</td>
<td>0.08</td>
<td>0.14</td>
</tr>
<tr>
<td>9</td>
<td>0.08</td>
<td>0.15</td>
</tr>
<tr>
<td>10</td>
<td>0.11</td>
<td>0.18</td>
</tr>
</tbody>
</table>

In comparing the results in Table 6.5 to those in Tables 6.1 and 6.2 for the same experiments, it is obvious that the GH program runs much faster, and the computation time does not increase as quickly with the increasing values of $n$ and $k$. We also compare the computation times of the GH program for the general case, which again involves a more complex scheduling procedure because the constraint of unit processing times is removed. Table 6.6 presents the computation times in seconds of the GH for the general case with parameters matching those of the experiments presented in Table 6.3.
Again, comparing the results in Table 6.6 to those in Table 6.3 illustrates that the GH program for the general case runs much faster than the CE program and the computation time does not increase as quickly with increasing size parameters. We also conduct experiments to demonstrate how lower $w_T$ values affect the GH program using parameters similar to those in Table 6.4. The computation times in seconds from these experiments are presented in Table 6.7.

Table 6.7. Computation Time of GH for General Case under Various $w_T$ Values

<table>
<thead>
<tr>
<th>$w_T$</th>
<th>Computation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.99</td>
<td>0.26</td>
</tr>
<tr>
<td>0.60</td>
<td>0.85</td>
</tr>
<tr>
<td>0.40</td>
<td>1.22</td>
</tr>
</tbody>
</table>

It is important to note that the value of $w_T$ does not affect the computation time of the GH program as much as in the CE program. There is a pattern of increase, but the results support the conclusion that the greedy heuristic is a polynomial time algorithm and can be used to solve large-scale problems with a wide range of parameters such as $w_T$ and due times. This includes many problems that cannot be solved or whose solution would be...
intractable using the CE method. To run the GH program for larger size problems we generate random inputs in a fashion similar to the preceding experiments. The due times are set to $n$ so that the portion of containers being inspected is roughly the same as in the preceding experiments. Table 6.8 presents the computation time of the GH for the general case in problems with larger numbers of containers in three stations with $wT=0.99$.

Table 6.8. Computation Time of GH for General Case with $k = 3$, $dc = n$

<table>
<thead>
<tr>
<th>$n$</th>
<th>Computation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.9</td>
</tr>
<tr>
<td>9</td>
<td>1.6</td>
</tr>
<tr>
<td>10</td>
<td>3.1</td>
</tr>
<tr>
<td>11</td>
<td>5.5</td>
</tr>
<tr>
<td>12</td>
<td>9.5</td>
</tr>
<tr>
<td>13</td>
<td>18.4</td>
</tr>
<tr>
<td>14</td>
<td>33.4</td>
</tr>
<tr>
<td>15</td>
<td>75.6</td>
</tr>
</tbody>
</table>

The results from Table 6.8 do show a pattern of increase in the times. The GH for special case problems runs much faster because the scheduling procedure is less complex. We conduct further experiments with the GH for the special case. Table 6.9 presents the computation time in seconds for experiments with varying problem sizes, due times equal to $n$ and $wT=0.99$. The information for experiments with three and four stations is also presented in graphical form in Fig. 6.2.
Table 6.9. Computation Time of GH for Special Case Large Problems

<table>
<thead>
<tr>
<th>n</th>
<th>k = 3</th>
<th>k = 4</th>
<th>k = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>20</td>
<td>0.5</td>
<td>1.0</td>
<td>1.7</td>
</tr>
<tr>
<td>30</td>
<td>1.4</td>
<td>2.8</td>
<td>5.1</td>
</tr>
<tr>
<td>40</td>
<td>3.2</td>
<td>6.6</td>
<td>12.2</td>
</tr>
<tr>
<td>50</td>
<td>6.3</td>
<td>13.6</td>
<td>25.2</td>
</tr>
<tr>
<td>60</td>
<td>11.6</td>
<td>25.2</td>
<td>65.1</td>
</tr>
<tr>
<td>70</td>
<td>20.5</td>
<td>45.9</td>
<td>87.5</td>
</tr>
<tr>
<td>80</td>
<td>33.9</td>
<td>77.0</td>
<td>146.9</td>
</tr>
<tr>
<td>90</td>
<td>53.8</td>
<td>123.2</td>
<td>281.7</td>
</tr>
<tr>
<td>100</td>
<td>81.8</td>
<td>189.9</td>
<td>365.2</td>
</tr>
</tbody>
</table>

Figure 6.2. Graph of Computation Times of GH for Special Case Large Problems
Table 6.9 and Fig. 6.2 both illustrate the pattern of increasing computation times with increasing problem size parameters $n$ and $k$. However, the pattern of increase and the actual times are still much smaller than those demonstrated in the CE program. It is concluded that the GH is capable of solving large scale problems that the CE could not feasibly perform.

6.5.2 Solution Performance Comparison

The complete enumeration and greedy heuristic approaches are tested on problems with the same parameters ($n$, $k$, and $wT$) and identical input values (risk profiles, due times, and processing times) to validate the performance of the heuristic. The objective function value, which includes contributions from the scaled risk and scaled tardiness, is guaranteed to be optimal in the solution found by complete enumeration, so the GH may perform as well but not better. The objective function value of the greedy heuristic solution is compared to the optimal objective function value from complete enumeration and the percent difference from optimal is presented in the following tables. The percent difference is calculated as $\frac{z(\text{GH}) - z(\text{CE})}{z(\text{CE})} \times 100$. For all examples with $wT=0.99$, which represents scenarios where the penalty for tardiness is very high relative to risk, the optimal tardiness is always found to be 0. Table 6.10 presents the results of the comparison for experiments using equal due times for the special case problem with three stations and $wT=0.99$. Table 6.11 presents results for similar experiments in a four station system.
Table 6.10. GH Solution Performance for Special Case with $k = 3$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$dc = 2$</th>
<th>$dc = 3$</th>
<th>$dc = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0.81</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.11. GH Solution Performance for Special Case with $k = 4$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$dc = 2$</th>
<th>$dc = 3$</th>
<th>$dc = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4.48</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.35</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.27</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table 6.10 it is observed that the solutions in the set of experiments described are within 1% of the true optimal objective value, and within 5% for Table 6.11. Further experiments are carried out with mixed due times and various $w_T$ values to assess the performance of the GH under these different conditions. The results from experiments with mixed due times and $w_T=0.99$ are presented in Table 6.12 for the special case problem in a three station system. The due code describes the generation of due times as follows: $dc=23$ indicates that alternating due times of 2 and 3 are generated for containers as needed. To demonstrate the effect of varying $w_T$, the other parameters are fixed at
three stations and seven containers with due times equal to 3. The results from these experiments of special case problems with varying $w_T$ are presented in Table 6.13.

Table 6.12. GH Solution Performance for Special Case under Mixed Due Times

<table>
<thead>
<tr>
<th>$n$</th>
<th>$dc = 23$</th>
<th>$dc = 24$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>4.94</td>
</tr>
</tbody>
</table>

Table 6.13. GH Solution Performance for Special Case under Various $w_T$ Values

<table>
<thead>
<tr>
<th>$w_T$</th>
<th>% difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.80</td>
<td>0</td>
</tr>
<tr>
<td>0.78</td>
<td>0</td>
</tr>
<tr>
<td>0.76</td>
<td>0</td>
</tr>
<tr>
<td>0.74</td>
<td>0</td>
</tr>
<tr>
<td>0.72</td>
<td>0.76</td>
</tr>
<tr>
<td>0.70</td>
<td>0</td>
</tr>
</tbody>
</table>

Similar results of near-optimal solutions are found in Tables 6.12 and 6.13, with solutions within 5% of optimal for the mixed due time experiments in Table 6.12 and within 1% for the experiments varying $w_T$ in Table 6.13. In the optimal solutions for the examples in Table 6.13, the tardiness values range from 0 to 3. The average percent difference is 0.3% over all examples (same due varying $k$, mixed due, and various $w_T$) presented for the special case.
Next general case examples with unequal processing times are considered. Again, for all examples with \( wT = 0.99 \), the optimal tardiness is found to be 0. Results of experiments for general case problems using equal due times for three and four stations with \( wT = 0.99 \) are presented in Table 6.14 and 6.15, respectively.

Table 6.14. GH Solution Performance for General Case with \( k = 3 \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( dc = 6 )</th>
<th>( dc = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2.33</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>11.72</td>
</tr>
<tr>
<td>8</td>
<td>1.06</td>
<td>9.58</td>
</tr>
</tbody>
</table>

Table 6.15. GH Solution Performance for General Case with \( k = 4 \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( dc = 6 )</th>
<th>( dc = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1.17</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Results from Table 6.14 and 6.15 demonstrate that the GH for general case problems is able to find the optimal solution in some experiments and in others returns a feasible solution that is less than 12% different from the optimal, for \( k=3 \) experiments in Table 6.14, and less than 2% different from optimal, for \( k=4 \) experiments in Table 6.15. Finally, experiments with mixed due times and various \( wT \) values are carried out to observe the heuristic performance under these conditions. The results from experiments with mixed due times in a three station system with \( wT = 0.99 \) for the general case are
presented in Table 6.16. For the examples demonstrating the effect of varying \( wT \), fixed parameter values of three stations and five containers with due times equal to 6 are chosen for clarity of presentation. Table 6.17 presents results from experiments for the general case problem with these fixed parameters under various \( wT \) values.

Table 6.16. GH Solution Performance for General Case under Mixed Due Times

<table>
<thead>
<tr>
<th>( n )</th>
<th>( dc = 47 )</th>
<th>( dc = 7654 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>13.85</td>
</tr>
<tr>
<td>6</td>
<td>0.41</td>
<td>0.58</td>
</tr>
<tr>
<td>7</td>
<td>10.68</td>
<td>1.09</td>
</tr>
<tr>
<td>8</td>
<td>3.46</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.17. GH Solution Performance for General Case under Various \( wT \) Values

<table>
<thead>
<tr>
<th>( wT )</th>
<th>( dc = 6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.60</td>
<td>11.42</td>
</tr>
<tr>
<td>0.56</td>
<td>9.66</td>
</tr>
<tr>
<td>0.52</td>
<td>5.74</td>
</tr>
<tr>
<td>0.48</td>
<td>1.53</td>
</tr>
<tr>
<td>0.44</td>
<td>7.74</td>
</tr>
</tbody>
</table>

Results in Table 6.16 and 6.17 respectively demonstrate the GH is capable of finding solutions within 14% of optimal for experiments with mixed due times, and within 12% for experiments with various \( wT \) values. The percent difference compares the total objective \((z)\) values, so in the case in Table 6.16 with five containers and \( dc = 7654 \) (which is the highest percent difference in all examples) there is an actual difference of
0.0011 in the risk objective. In comparing the solutions returned by the two methods it is observed that the CE returns a schedule with five containers inspected at a particular station, while the best allocation and schedule created by the GH is identical except it has only four containers being inspected at that station. The average percent difference from optimal over all examples (same due varying $k$, mixed due, and various $wT$ values) is 3.4% for the general case. It is observed that the GH doesn’t perform as well with mixed due dates and varying $wT$ values but still returns close to optimal solutions. The heuristic may be modified to improve performance. The results from experiments presented in this section support the conclusion that the greedy heuristic returns near-optimal solutions for the experiments presented in this paper, and will likely return near-optimal solutions for large-scale problems as well in a much shorter computation time.

6.6 Discussion

In this chapter we consider the container inspection problem with multiple risk categories and develop a new mathematical model to obtain the optimum allocation of containers to inspection stations and the optimal schedule of inspection operations at each station. The model is different from those presented in previous chapters because the basic assumptions of container contents and acceptability are modified. We assume detailed intelligence information which allows a more fitted inspection plan for each container. Planning effective inspection while considering port operations and the need to keep container flow moving proves to be an important problem.
The performance of an inspection policy relative to each risk category can be considered separately, resulting in a multi-objective problem. The optimization can also be modified to use weight $w_j$ for the $j^{th}$ risk and compute a weighted sum over risk categories of cost or another objective. There are multiple objectives to be considered, for example the objectives we have considered in previous chapters: PFA, PFR, cost, and time. Solution methods for problems with additional constraints may be developed. One could also explore the formulation and optimization of this problem when taking into account measurement error and allowing for repeat inspections.

The numerical examples in this chapter demonstrate that a complete enumeration method quickly becomes infeasible to run for increasing numbers of containers or stations. However, the developed greedy algorithm can provide solutions to large-scale problems in a relatively short computation time. The time required to obtain a solution is an important aspect of the problem, as a solution must be run for a new set of containers and data every day, or every dedicated inspection period. The proposed heuristic also returns near-optimal solutions for the experiments presented.
CHAPTER 7

CONCLUSIONS

The container inspection problem is becoming increasingly important with growing security concerns and the rising volume of international trade. In this dissertation we have proposed different formulations of mathematical models in order to evaluate and improve the inspection process. The formulations consider different system definitions, risk information, and inspection policies. Several objectives are pursued, including probability of false acceptance and rejection, cost and time of inspection, and container tardiness. This work represents a unique mathematical formulation of the port-of-entry container inspection problem. We consider four variations of the container inspection problem described as follows.

We model an inspection system with a Boolean decision function combining station results, define an optimal sequence with respect to the expected cost of inspection, and apply it in the optimization of sensors’ threshold levels at inspection stations and the sequence of stations to achieve the minimum expected total cost of misclassification errors and inspection. It is found that inspection sequence and the threshold levels of the sensors are critical in the decision process of accepting or classifying a container as suspicious. The results demonstrate the inherent tradeoff between comprehensive inspection and misclassification and provide a solution method for the sequence problem.
and overall policy optimization for systems in series, parallel, parallel-series, and series-parallel.

Next we expand the objective to include the minimization of the total expected time a container spends in the inspection system, and develop a multi-objective optimization approach to determine the optimal inspection policy parameters. The theorems for finding the optimal sequence of inspections for expected cost are modified to find an optimal sequence that considers cost and time simultaneously. The results identify factors that are important in the design of a system and provide easy to apply methods for determining an optimal inspection policy.

We introduce a measurement error term that is independent of the natural variation in container attribute measurements and investigate the impact of such errors on the system performance. The results of a numerical example illustrate the effects of varying measurement error on the performance of a single inspection station across several threshold levels. We then propose a repeat inspection procedure and analyze its effect on the station performance. The results of an example with varying threshold and re-inspection band parameters demonstrate the benefit in decreased misclassifications from the extra step of inspection and suggest that the repeat inspection procedure is a valid approach to reducing the impact of measurement errors on the performance of the inspection system. We also define three objectives of a repeat inspection policy and discuss novel solution methods for this multi-objective problem.
Finally we consider multiple independent risk categories and determine the optimal allocation and scheduling of container inspection operations to minimize false acceptance and tardiness objectives. The theoretical contributions of this research include a unique approach to scheduling container inspections. We introduce a new class of problems in scheduling theory in which the allocation of inspections (or operations) is not defined and appears as a decision variable in the solution. A solution approach to this simultaneous allocation and scheduling problem is proposed. The performance of this greedy heuristic is compared with a complete enumeration method in several experiments with varying problem parameters. Results indicate that the complete enumeration approach quickly becomes unmanageable as the problem size increases, however the proposed heuristic returns near-optimal solutions and can quickly solve large-scale problems.

It is important to note that the methodologies presented in this dissertation are not tied to or limited by current sensor technology and can be easily adapted to future capabilities. The ability to extract knowledge concerning container contents and the interpretation and availability of intelligence information used to characterize incoming container risk profiles are two major factors in container inspection that will continue to develop and improve with future research. It is expected that these advances will improve the outcome of inspection, or at least keep pace with the technologies employed to smuggle unwanted cargo.

This research focuses on the optimization of an inspection system given the current environment and parameters and addresses specifically defined inspection systems. The
methodology could be extended to different kinds of systems. For example, the determination of the optimal sequence and threshold levels of sensors arranged in non-standard arrangements such as a general network of sensors or $k$-out-of-$n$ arrangement warrants further investigation. Other extensions of this work could also include the definition of several versions of inspection policies involving repeat inspections and a comparison of their performances. In the allocation and scheduling problem several different objectives may be considered as mentioned in Chapter 6 and repeat inspection may be introduced as a potential solution component, thereby expanding the solution space.
References


Appendix A

A.1 Induction Formula for the Parallel System

Suppose the \((k + 1)^{th}\) sensor is added to the \(k\)-parallel sensors with \(c_1/p_1 \leq \ldots \leq c_k/p_k \leq c_{k+1}/p_{k+1}\). For this system of \((k+1)\) parallel sensors, the cost of inspection can be calculated by the following induction formula: \(C_i^{(1)} = c_i\) and

\[
C_i^{(k+1)} = C_i^{(k)} + c_{k+1} \sum_{j=1}^{k} q_j \quad \text{for } k = 1, 2, \ldots
\]

The cost function of making a false decision is \(C_{r_i}^{(k+1)} = \pi c_{r_i} (1 - A_{r_i}^{(k+1)}) + (1 - \pi) c_{r_{FR}} B_{r_i}^{(k+1)}\), where \(A_{r_i}^{(k+1)}\) and \(B_{r_i}^{(k+1)}\) can be computed by the following induction formulas:

\[
A_{r_i}^{(1)} = \Phi \left( \frac{T_{i1} - 1}{\sigma_{i1}} \right), \quad B_{r_i}^{(1)} = \Phi \left( \frac{T_{i1}}{\sigma_{i0}} \right), \quad A_{r_i}^{(k+1)} = A_{r_i}^{(k)} \Phi \left( \frac{T_{r_{i,k+1}} - 1}{\sigma_{r_{i,k+1}}} \right), \quad \text{and} \quad B_{r_i}^{(k+1)} = B_{r_i}^{(k)} \Phi \left( \frac{T_{r_{i,k+1}}}{\sigma_{r_{i,k+1}}} \right) \quad \text{for}\]

\(k = 1, 2, \ldots\) where \(\Phi = 1 - \Phi\).

A.2 Induction Formula for the Series System

Suppose the \((k + 1)^{th}\) sensor is added to the \(k\)-series sensors with \(c_1/q_1 \leq \ldots \leq c_k/q_k \leq c_{k+1}/q_{k+1}\). For this system of \((k+1)\) series sensors, the cost of inspection can be calculated by the following induction formula: \(C_i^{(1)} = c_i\) and
\[ C_{i}^{(k+1)} = C_{i}^{(k)} + c_{k+1} \prod_{j=1}^{k} p_j \text{ for } k = 1, 2, \ldots \]

The cost function of making a false decision is
\[ C_{fa}^{(k+1)} = \pi c_{fa} A^{(k+1)} + (1-\pi) c_{fr} (1 - B^{(k+1)}) \],
where \( A^{(k+1)} \) and \( B^{(k+1)} \) can be computed by the following induction formulas:
\[ A^{[i]} = \Phi \left( \frac{T_i - 1}{\sigma_{i1}} \right), \quad B^{[i]} = \Phi \left( \frac{T_i}{\sigma_{01}} \right), \quad A^{[k+1]} = A^{[k]} \Phi \left( \frac{T_{k+1} - 1}{\sigma_{r,k+1}} \right), \quad \text{and} \quad B^{[k+1]} = B^{[k]} \Phi \left( \frac{T_{k+1}}{\sigma_{0,k+1}} \right) \]
for \( k = 1, 2, \ldots \).

**A.3 Induction Formula for the Parallel-Series System**

**Step 1:** Add \((m+1)^{th}\) sensor to each branch: \((n, m)\rightarrow(n, m+1)\)

If \( c_{i,1} / q_{i,1} \leq \ldots \leq c_{i,m} / q_{i,m} \leq c_{i,m+1} / q_{i,m+1} \) for all branches, Section 3.2.2 gives the minimum inspection cost as
\[ C_{i}^{(m+1)} = C_{i}^{(m+1)} + \sum_{i=2}^{m} C_{i,j}^{(m+1)} \prod_{j=1}^{i-1} \prod_{k=1}^{m+1} (1 - p_{j,k}) \],
where the inspection cost of the \( i^{th} \) path of \( m+1 \) sensors in series \( C_{i,i}^{(m+1)} \), for \( i = 1, 2, \ldots, n \), can be calculated by the induction formula in A.2.

From the results from Section 3.2.1, the total expected cost of misclassification is
\[ C_{fa}^{(m+1)} = \pi c_{fa} [1 - PM^{(m+1)}] + (1-\pi) c_{fr} PN^{(m+1)}, \quad \text{where} \quad PM^{(m+1)} = \prod_{i=1}^{n} (1 - M_{i}^{(m+1)}), \]
\[ PN^{(m+1)} = \prod_{i=1}^{n} (1 - N_{i}^{(m+1)}), \quad \text{and} \quad M_{i}^{(m+1)} \text{ and } N_{i}^{(m+1)} \text{ can be updated by the induction} \]
formulas $M_i^{[m+1]} = M_i^{[m]} \Phi \left( \frac{T_{i,m+1} - 1}{\sigma_{1,(i,m+1)}} \right)$ and $N_i^{[m+1]} = N_i^{[m]} \Phi \left( \frac{T_{i,m+1}}{\sigma_{0,(i,m+1)}} \right)$.

**Step 2:** Add the \((n+1)\)th branch with \(m\) sensors: \((n, m) \to (n+1, m)\)

If \(C_{1,1}/Q_1 \leq \ldots \leq C_{1,n}/Q_n \leq C_{1,n+1}/Q_{n+1}\), Section 3.2.2 gives the minimum inspection cost as: $C_i^{[n+1]} = C_i^{[n]} + C_{i,n+1} G_n$ where \(G_n\) can be updated by induction: \(G_1 = Q_1\) and $G_{j+1} = G_j Q_j$ with $Q_j = 1 - \prod_{j=1}^{m} [(1 - \pi) \Phi \left( \frac{T_j}{\sigma_{o,j}} \right) + \pi \Phi \left( \frac{T_j - 1}{\sigma_{1,j}} \right)]$.

From the result of Section 3.2.1, the total expected cost of misclassification is $C_F^{[n+1]} = \pi c_{p} A_1 \left[ 1 - PA^{[n]} A_{n+1} \right] + (1 - \pi) c_{p} B_{n+1}$ where \(PA^{[n+1]}\) and \(PB^{[n+1]}\) can be updated by the induction formulas: \(PA^{[1]} = A_1\), \(PA^{[j+1]} = PA^{[j]} A_j\), \(PB^{[1]} = B_1\), \(PB^{[j+1]} = PB^{[j]} B_j\) for \(j = 1, 2, \ldots, n\) and \(A_i = 1 - \prod_{j=1}^{m} \Phi \left( \frac{T_{ij} - 1}{\sigma_{1,ij}} \right)\) and \(B_i = 1 - \prod_{j=1}^{m} \Phi \left( \frac{T_{ij}}{\sigma_{o,ij}} \right)\) for \(j = 1, 2, \ldots, n + 1\).

**A.4 Induction Formula for the Series-Parallel System**

**Step 1:** Add the \((m+1)\)th sensor in each subsystem: \((n, m) \to (n, m+1)\)

If \(c_{i,1} / p_{i,1} \leq \ldots \leq c_{i,m} / p_{i,m} \leq c_{i,m+1} / p_{i,m+1}\) for all the branches, Section 3.2.2 gives the minimum inspection cost as $C_i^{[n,m+1]} = C_i^{[m+1]} + \sum_{i=2}^{n} C_i^{[m+1]} \prod_{j=1}^{n} [1 - \prod_{j=1}^{m} (1 - p_{j,i})]$ where the inspection cost of the \(i\)th path of \(m+1\) sensors in series $C_i^{[m+1]}$, for \(i = 1, 2, \ldots, n\), can be calculated by the induction formula in A.1.
From the result of Section 3.2.1, the total expected cost of misclassification is
\[ C_F^{[m+1]} = \pi c_{FA}(PA^{[m+1]}) + (1 - \pi)c_{FR}(1 - PB^{[m+1]}) \]
where \( PA^{[m+1]} = \prod_{i=1}^{n} (1 - A_i^{[m+1]}) \),
\[ PB^{[m+1]} = \prod_{i=1}^{n} (1 - B_i^{[m+1]}) \], and

\[ A_i^{[m+1]} \quad \text{and} \quad B_i^{[m+1]} \]
can be updated by the induction formulas
\[ A_i^{[m+1]} = A_i^{[n]} \Phi \left( \frac{T_{i,n+1} - 1}{\sigma_{i,n+1}} \right) \],
\[ B_i^{[m+1]} = B_i^{[n]} \Phi \left( \frac{T_{i,n+1} - 1}{\sigma_{i,n+1}} \right) \], where \( \Phi = 1 - \Phi \).

**Step 2:** Add the \((n+1)\)th subsystem with \( m \) sensors: \((n, m) \rightarrow (n+1, m)\)

If \( C_{I,1}/Q_1 \leq ... \leq C_{I,n}/Q_n \leq C_{I,n+1}/Q_{n+1} \), Section 3.2.2 gives the minimum inspection cost as
\[ C_I^{[n+1]} = C_I^{[n]} + C_{I,n+1}G_n \]
where \( G_n \) can be updated by induction: \( G_1 = P \) and
\[ G_{j+1} = G_j P_j \]
with \( P_j = 1 - \prod_{s=1}^{m} [(1 - \pi)\Phi \left( \frac{T_{js} - 1}{\sigma_{o,js}} \right) + \pi \Phi \left( \frac{T_{js} - 1}{\sigma_{1,js}} \right)] \).

From the result of Section 3.2.1, the total expected cost of misclassification is
\[ C_F^{[n+1]} = \pi c_{FA}(PA^{[n]})A_{n+1} + (1 - \pi)c_{FR}PB^{[n]}B_{n+1} \]
where \( PA^{[n+1]} \) and \( PB^{[n+1]} \) can be updated by the induction formulas:
\( PA^{[1]} = A_1 \), \( PA^{[j+1]} = PA^{[j]}A_j \), \( PB^{[1]} = B_1 \), \( PB^{[j+1]} = PB^{[j]}B_j \)
for \( j = 1, 2, ..., n \) and \( A_i = 1 - \prod_{j=1}^{m} \Phi \left( \frac{T_{ij} - 1}{\sigma_{1,ij}} \right) \), \( B_i = 1 - \prod_{j=1}^{m} \Phi \left( \frac{T_{ij} - 1}{\sigma_{o,ij}} \right) \) for \( j = 1, 2, ..., n+1 \), where \( \Phi = 1 - \Phi \).
Appendix B

Proof of Theorem 1(a) in Chapter 4:

For a series Boolean decision function, the fitness function given the set of weights \((w_1, w_2)\) is:

\[
f_{w_1,w_2}(S,T) = w_1c_{\text{total}} + w_2t_{\text{total}} = w_1(C_J + C_F) + w_2t_{\text{total}}
\]
\[
= w_1C_J + w_2t_{\text{total}} + w_1C_F \tag{B.1}
\]

where

\[
w_1C_F = w_1\left[\pi PFA c_{FA} + (1 - \pi) PFR c_{FR}\right] = w_1\left[\pi \left(\Phi\left(\frac{T_i - 1}{\sigma_i}\right)\right)c_{FA} + (1 - \pi)\left(1 - \Phi\left(\frac{T_i}{\sigma_{0i}}\right)\right)c_{FR}\right] \tag{B.2}
\]

and

\[
w_1C_J + w_2t_{\text{total}} = w_1\left[c_i + \sum_{i=2}^{n} \prod_{j=1}^{i-1} p_j c_i\right] + w_2\left[t_i + \sum_{i=2}^{n} \prod_{j=1}^{i-1} p_j t_i\right]
\]
\[
= (w_1c_i + w_2t_i) + \sum_{i=2}^{n} \prod_{j=1}^{i-1} p_j (w_1c_i + w_2t_i) \tag{B.3}
\]

It is obvious that Eq. (B.2) does not depend on the sequence \(S\). And by PROPOSITION 1

of Butterworth (1972), the sequential order is optimum, in the sense of minimizing Eq. (B.3) for the given set of weights \((w_1, w_2)\) and a given set of threshold values, if and only

if \(w_1c_i + w_2t_i) / q_i \leq (w_1c_2 + w_2t_2) / q_2 \leq \ldots \leq (w_1c_n + w_2t_n) / q_n\).
Theorem 1(b) in Chapter 4 can be proved by similar argument and PROPOSITION 2 of Butterworth (1972) and Theorem 2 can be proved by similar argument and THEOREM 3.1 and 3.2 of Ben-Dov (1981).
## Appendix C

Table of 24 experiment designs from Chapter 4 Section 4.4

<table>
<thead>
<tr>
<th>Design</th>
<th>$\pi$</th>
<th>${\sigma_0,\sigma_1}$</th>
<th>$c_{FA}$</th>
<th>$c_{FR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+05</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+05</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+07</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+07</td>
<td>400</td>
</tr>
<tr>
<td>5</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+05</td>
<td>200</td>
</tr>
<tr>
<td>6</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+05</td>
<td>400</td>
</tr>
<tr>
<td>7</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+07</td>
<td>200</td>
</tr>
<tr>
<td>8</td>
<td>2E-04</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+07</td>
<td>400</td>
</tr>
<tr>
<td>9</td>
<td>2E-04</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+05</td>
<td>200</td>
</tr>
<tr>
<td>10</td>
<td>2E-04</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+05</td>
<td>400</td>
</tr>
<tr>
<td>11</td>
<td>2E-04</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+07</td>
<td>200</td>
</tr>
<tr>
<td>12</td>
<td>2E-04</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+07</td>
<td>400</td>
</tr>
<tr>
<td>13</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+05</td>
<td>200</td>
</tr>
<tr>
<td>14</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+05</td>
<td>400</td>
</tr>
<tr>
<td>15</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+07</td>
<td>200</td>
</tr>
<tr>
<td>16</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.25], [0.25 \ 0.15 \ 0.15]]$</td>
<td>1E+07</td>
<td>400</td>
</tr>
<tr>
<td>17</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+05</td>
<td>200</td>
</tr>
<tr>
<td>18</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+05</td>
<td>400</td>
</tr>
<tr>
<td>19</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+07</td>
<td>200</td>
</tr>
<tr>
<td>20</td>
<td>5E-05</td>
<td>$[[0.15 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.15]]$</td>
<td>1E+07</td>
<td>400</td>
</tr>
<tr>
<td>21</td>
<td>5E-05</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+05</td>
<td>200</td>
</tr>
<tr>
<td>22</td>
<td>5E-05</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+05</td>
<td>400</td>
</tr>
<tr>
<td>23</td>
<td>5E-05</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+07</td>
<td>200</td>
</tr>
<tr>
<td>24</td>
<td>5E-05</td>
<td>$[[0.25 \ 0.25 \ 0.35], [0.35 \ 0.25 \ 0.25]]$</td>
<td>1E+07</td>
<td>400</td>
</tr>
</tbody>
</table>
Appendix D

MATLAB functions for greedy search and scheduling heuristics for special case problems

function \[RRe, Te, sopte, ze\] = GHp1(Risk, PTFR, wT, due)
% File GHp1.m
% Runs greedy heuristic for k stations, n containers
% Assumes pt(i,j)==1, due times can be different
% returns the residual risk (RR), tardiness (T), schedule (s)
% input: weight of tardiness (wT) and due times (due)
% output: RR,T, sopt, zopt

epss=[.0001, .0005, .001];
n=size(Risk,1);
k=size(Risk,2);
y=zeros(size(Risk));
yc=y;
ze=2;
[Tmx, Rmx, Rmn]=Scaling(Risk, PTFR, due);

for a=1:size(epss,2)
    % loop through epsi values for best overall 'ze'
    epsi=epss(a);
    z=2;
    don=0;
    y=0*y;
    potential=Risk;
    for i=1:n
        potential(i,:)=Risk(i,:).*PTFR(1,:);
    end
    %Determine Best Allocation
    while don==0
        %Start loop, repeat until all slots are full or potential=0
        [ap, bp]=find(potential > max(0, max(potential(:))-epsi));
        yc=yc*0;
        ze=0;
        for g=1:size(ap,1)
            %Consider ctnr:ap(g), stn:bp(g) with large potential
            yf=y;
            yf(ap(g),bp(g))=1;
            %check implications
        end
        don=1;
    end
end
yc(:,:,g)=yf;
PRes=rescalc(Risk,PTFR,yf);
RRc(g)=sum(sum(PRes));
LB(g)=LoB(yf,due);
zc(g)=(RRc(g)-Rmn)*(1-wT)/(Rmx-Rmn)+ LB(g)*wT/Tmx;
end
win=0;
bf=0;
 [~,ind]=sort(zc);
for f=1:size(ap,1)
    %ind(f) is the location of the f lowest zc value
    if win==0 && zc(ind(f))<z
        yf=yc(:,:,ind(f));
        [s,Ta]=Scheduler(yf,due);
        if Ta<LB(ind(f))
            return
        end
        za=(RRc(ind(f)-Rmn)*(1-wT)/(Rmx-Rmn)+ Ta*wT/Tmx;
        if f==size(ap,1)
            if za<z
                win=1;
                %winner, update opt solution
                z=za;
                y=yf;
                sopt=s;
                T=Ta;
                RR=RRc(ind(f));
                potential(ap(ind(f)),bp(ind(f)))=0;
            end
        elseif za< min(z,zc(ind(f+1)))
            if za<z
                win=1;
                z=za;
                y=yf;
                sopt=s;
                T=Ta;
                RR=RRc(ind(f));
                potential(ap(ind(f)),bp(ind(f)))=0;
            elseif za<z
                % best so far, reserve for future comparison
                z=za;
                bf=f;
                Tf=Ta;
                sf=s;
            end
        end
    end
end
if win==0
if bf==0
  \% none of ap,bp usefull
  for g=1:size(ap,1)
    potential(ap(g),bp(g))=0;
  end
else
  \% appoint bf as opt
  y=yc(:,:,\text{ind(bf)});
  RR=RRc(\text{ind(bf)});
  s\text{opt}=sf;
  T=Tf;
  potential(ap(\text{ind(bf)}),bp(\text{ind(bf)}))=0;
end
end
\%check ending conditions
if sum(sum(potential))==0
  don=1;
end
end
if z<ze
  ze=z;
  s\text{opte}=s\text{opt};
  T\text{e}=T;
  R\text{Re}=R\text{R};
end
end

function \[s_0,T\text{min}\] = Scheduler\(y,\text{due}\)
\% File Scheduler.m
\% Runs GH Scheduler for k stations, n containers
\% Assumes pt\((i,j)\)==1, due times can be different
\% returns the best schedule and tardiness (T)
\% input: allocation (y) and due times (due)

\text{n=size(y,1);}
\text{k=size(y,2);}
\text{T\text{min}=0;}
\text{s=\text{zeros(max(due),k);} }
\text{sched=\text{zeros(1,k);} }
\text{R\text{max}=3;} \%max \#rounds to move i(Ti>1) and reschedule
\text{bk=\text{zeros(k,size(s,1),n);} }
LB=LoB(y,due);
Rnd=1;

for i=1:n
    for j=1:k
        bk(j,due(i),i)=i*y(i,j);
    end
end
while Rnd<=Rmax
    %repeat future rounds from here
s=zeros(max(due)+LB,k);
trd=zeros(size(due));
tbs=y;
nbt=zeros(1,k);
nbb=zeros(1,k);
prty=max(0,sum(y,2)-due+1);
sched=sched*0;
ct=1;   %First row
while sum(sum(tbs))>0  %Loop: rows of s until done
    %Beginning of each row
    for j=1:k  %next block time, business, and contents
        if sum(tbs(:,j))==0
            sched(j)=1;
        else
            nbt(j)=find(sum(bk(j,:,:),3)>0,1);
            nbb(j)=(size(find(bk(j,nbt(j),:)),1)>=nbt(j)-ct+1);
            nbc(:,j)=bk(j,nbt(j),:);
        end
    end
    for a=1:k   %Loop through k stations
        %choose station
        j=0;
        minj=min(n+1,prty);  %min# in nbc
        maxt=0;   %to break ties: sum(tbs(:,j))
        if sum((sched<1).*nbb) %unsched stations w/ nbb
            unbb=find((sched<1).*nbb);
        else
            unbb=find(sched<1);
        end
        for b=1:size(unbb,2)
            if size(find(nbc(:,unbb(b))),1)<minj
                j=unbb(b);
                minj=size(find(nbc(:,j)),1);
                maxt=sum(tbs(:,j));
            elseif size(find(nbc(:,unbb(b))),1)==minj
                %do something
            end
        end
    end
end
if sum(tbs(:,unbb(b)))>maxt
    maxt=sum(tbs(:,unbb(b)));    
    j=unbb(b);                  
end
end

if j>0
    mp=-1;
    maxt=0;
    unbb=find(nbc(:,j));
    for b=1:size(unbb,1)
        [~,c]=find(bk(:,:,unbb(b))==unbb(b),1);
        busyj=[];  %set busy stations
        sb=0;
        for j1=1:k
            if size(find(sum(bk(j1,1:c,:))),1)>=c-ct-
sched(j1)+1
                sb=sb+1;
                busyj(sb)=j1;
            end
        end
        if prty(unbb(b))>mp
            i=unbb(b);
            mp=prty(i);
            maxt=sum(tbs(i,busyj));
        elseif prty(unbb(b))==mp
            if sum(tbs(i,busyj))>maxt
                maxt=sum(tbs(i,busyj));
                i=unbb(b);
            end
        end
    end
end
s(ct,j)=i;
tbs(i,j)=0;
bk(j,:,i)=0*bk(j,:,i);
sched(j)=1;
unbb=find(sched<1);
for b=1:size(unbb,2)
    j1=unbb(b);
    nbc(i,j1)=0;
    if sum(nbc(:,j1))<1
        if sum(sum(bk(j1,nbt(j1)+1:end,:,3)))==0
            sched(j1)=1;
        else  %find next block
            nbt(j1)=find(sum(bk(j1,nbt(j1)+1:end,:,3)>0,1)+nbt(j1);
            nbb(j1)=(size(find(sum(bk(j1,1:nbt(j1,:)))))),1)~=nbt(j1)-ct+1);
nbc(:,j1)=bk(j1,nbt(j1,:));
    end
    end
    end
else
    sched=ones(1,k);
end
end
% all stations scheduled
sched=sched*0;
c=ct+1;
prty=max(max(sum(y,2)-due+1,sum(tbs,2)-due+ct),0);
end

% Calculate tardiness(i) and total
for i=1:n
    [c,d]=find(s==i);
    if isempty(c)
        trd(i)=0;
    else
        trd(i)=max(0,max(c)-due(i));
    end
end
RdTmin=sum(trd);
% compare results to best Tmin from previous rounds
if Rnd==1
    Tmin=RdTmin;
s0=s;
elseif RdTmin<Tmin
    Tmin=RdTmin;
s0=s;
end
% "so"=best sched and "Tmin"=best sum(trd)
mvi=0;
tardyj=[1:k];
if max(trd)>1 && Rnd<Rmax
    for b=max(trd):-1:2
        if mvi==0
            c=find(trd==b);
            maxt=1;
            for a=1:size(c,1)
                if sum(y(c(a),tardyj))> maxt
                    mvi=c(a);
                    maxt=sum(y(mvi,tardyj));
                end
            end

        end
    end
end
if mvi>0
    bk=zeros(k,size(s,1),n);
    for i=1:n
        for j=1:k
            bk(j,due(i),i)=i*y(i,j);
        end
    end
    for j=1:k
        bk(j,due(mvi),mvi)=0;
        bk(j,due(mvi)+1,mvi)=mvi*y(mvi,j);
        % move mvi to bk(due+1) for all yij
    end
else
    Rnd=Rmax+1;
end
Rnd=Rnd+1;
% repeat round
MATLAB functions for greedy search and scheduling heuristics for general case problems

function [RR1,T1,sopt1,sfo1,z1] = GHwt(Risk,PTFR,wT,due,t)
% File GHwt.m
% Runs greedy heuristic for k stations, n containers
% general case, no restriction on times
% returns the residual risk (RR), tardiness (T),
% container schedule (sopt), finish times (sfo), and z
% input: weight of tardiness (wT) and due times (due)
% output: RR, T, schedule, finish times, z=Rs*(1-wT)+Ts*wT

n=size(Risk,1);
k=size(Risk,2);
y=zeros(size(Risk));
don=0;
z1=5;
upd=0;

[Tmx, Rmx, Rmn]=Scalingwt(Risk, PTFR, due, t);
wA=[1,1,2,10]; % weights, first two differ by te
z=5*ones(length(wA));

for a=1:length(wA)
    % use trialcode to identify which trial
    % tc=num2str(wA(a));
    if a==1
        te=10;
    else
        te=1;
    end
    potential=Risk;
    for i=1:n
        potential(i,:)=Risk(i,:).*PTFR(1,:);
    end
    z(a)=5;
    don=0;
    y=zeros(size(Risk));
    while don==0
        [ma,b]=max(potential.^wA(a)./(t.^te));
        % code to update schedule and finish times
    end
end

Appendix E
\[ [\sim, c] = \max(\text{ma}) ; \]
\[ y(b(c), c) = 1 ; \]
\[ \text{PRes} = \text{rescalc} (\text{Risk, PTFR, y}) ; \]
\[ \text{RRo} = \sum(\sum(\text{PRes})) ; \]
\[ \text{TminLB} = \text{LBt} (y, \text{due}, t) ; \]
\[ zm = (\text{RRo} - \text{Rmn}) * (1 - wT) / (\text{Rmx} - \text{Rmn}) + wT * \text{TminLB} / \text{Tmx} ; \]
\[ \text{if} zm > z(a) \]
\[ Ta = \text{TminLB} ; \]
\[ \text{else} \]
\[ \%\text{subroutine} '\text{Schedwt}' \text{ returns min tardiness} \]
\[ [Ta, s, sf] = \text{Schedwt} (y, \text{due}, t, \text{TminLB}) ; \]
\[ \text{end} \]
\[ \text{if} ((\text{RRo} - \text{Rmn}) * (1 - wT) / (\text{Rmx} - \text{Rmn}) + wT * Ta / \text{Tmx}) < z(a) \]
\[ z(a) = (\text{RRo} - \text{Rmn}) * (1 - wT) / (\text{Rmx} - \text{Rmn}) + wT * Ta / \text{Tmx} ; \]
\[ \text{RR} = \text{RRo} ; \]
\[ T = Ta ; \]
\[ yopt = y ; \]
\[ sopt = s ; \]
\[ sfo = sf ; \]
\[ \text{potential}(b(c), c) = 0 ; \]
\[ \text{else} \]
\[ y(b(c), c) = 0 ; \]
\[ \text{potential}(b(c), c) = 0 ; \]
\[ \text{end} \]
\[ \text{if} \sum(\sum(\text{potential})) == 0 \]
\[ \text{don} = 1 ; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \%\text{compare to overall zo} \]
\[ \text{if} z(a) < z1 \]
\[ z1 = z(a) ; \]
\[ \text{RR1} = \text{RR} ; \]
\[ T1 = T ; \]
\[ sopt1 = sopt ; \]
\[ sfo1 = sf ; \]
\[ \text{upd} = \text{upd} + 1 ; \]
\[ \text{end} \]
\[ \text{end} \]

function \[ [\text{Tmin}, \text{sopt}, \text{sfopt}] = \text{Schedwt} (y, \text{due}, t, \text{LB}) \]

% File Schedwt.m
% Scheduler for GH (n,k) problems
% Builds and compares schedules to find the one that
% returns the minimum total tardiness
% input: allocation(y), due times, processing times, LB
% output: total tardiness(Tmin),
%         (n,k) matrices of container schedule and finish times

n=size(y,1);
k=size(y,2);
Tmin=10000;
Mtrd=10000;
s=zeros(n,k);
Seq=s;
s otp=s;
s fopt=s;

wA=0:0.2:1; % weights for seqA
wB=0:0.2:1; % weights for seqB
pk=perms(1:k);
spk=size(pk,1);

ty=t.*y;
tbar=sum(sum(ty))/sum(sum(ty>0));
dbar=sum(due)/sum(due>0);
tyb=sum(sum(ty))/sum(sum(ty,2)>0);
for j=1:k
    odj=0;
tj=ty(:,j);
sdue=due.*y(:,j);
[v,A]=sort(sdue);
A=A(v>0);
os=Sequence(0,A,due,tj);
    if isempty(os)==0
    c(1)=t(os(1),j);
odj=odj+max(0,c(1)-due(os(1)));
    for i=2:length(os)
c(i)=c(i-1)+t(os(i),j);
odj=odj+max(0,c(i)-due(os(i)));
    end
end
Seq(:,j)=[os;zeros(n-length(os),1)];
od(j)=odj;
end
% finds optseq for each station j according to Sequence algorithm

trials=1+length(wA)+length(wB);
for a=1:spk*trials
%use trialcode to identify which trial run, form s from sequences
if a<=spk
    %Use optseq above
tc='os';
s=Seq;
elseif a<=spk*(1+length(wA))
    %Use seqA
    wi=ceil((a-spk*(1))/spk);
    %tc=[num2str(wA(wi)) 'A'];
    for j=1:k
        [v,A]=sort(((1-wA(wi))/tbar)*ty(:,j)+(wA(wi)/dbar)*due.*y(:,j));
        A=A(v>0);
        s(:,j)=[A; zeros(n-length(A),1)];
    end
else
    %Use seqB
    wi=ceil((a-spk*(1+length(wA)))/spk);
    %tc=[num2str(wB(wi)) 'B'];
    for j=1:k
        [~,A]=sort(((1-wB(wi))/dbar)*due.*y(:,j)-(wB(wi)/tyb)*sum(ty,2).*y(:,j));
        A=A(y(A,j)>0);
        s(:,j)=[A; zeros(n-length(A),1)];
    end
end
end
% Determine station schedule order (sord):
sord=pk(mod(a,spk)+1,:);
tc=[tc num2str(mod(a,spk)+1)];

trd=zeros(n,1);
ft=trd;
sf=zeros(size(s));  %scheduled finish
tbs=s;
nav=zeros(1,k);     %next available time for j
for j=1:k
    nt(j)=t(max(1,s(1,j)),j);         %next tij for j=1:k
end
ic=ones(1,k);       % index of current slot in s to be scheduled

% Build time schedule
while sum(sum(tbs))>0
    for f=1:k   %each station considered once in each loop
        j=sord(f);
        %tj=ty(:,j);
        %sdue=due.*y(:,j);
        if tbs(1,j)>0

    end
end

% Check feasibility of tbs(1,j)
[e,c]=find(s==tbs(1,j));
e=e(c~=j);
c=c(c~=j);
flag=0;
gp=0;
for d=1:length(e)
if sf(e(d),c(d))>nav(j) && sf(e(d),c(d)) - t(tbs(1,j),c(d)) < nav(j) + nt(j)
    gp=max(gp,sf(e(d),c(d))-nav(j));
    flag=length(e);
end
end
while flag>0
    for d=1:length(e)
        if sf(e(d),c(d))>nav(j)+gp && sf(e(d),c(d)) - t(tbs(1,j),c(d)) < nav(j) + nt(j) + gp
            gp=sf(e(d),c(d))-nav(j);
            flag=length(e);
        else
            flag=flag-1;
        end
    end
end
if gp==0
    bunch=tbs(1,j);
dg=0;
else
    % Find bunch=[ctnrs], dg=[delays] with no conflicts
    bunch=[];
dg=[];
tbj=tbs((tbs(:,j)>0),j);
ct=nav(j);
i=tbj(1);
while isempty(bunch(bunch==i))
    gap=[];
    for b=1:length(tbj)
        % Schedule at ct, or closest available
        if b==1
            gap(1)=max(0,gp-(ct-nav(j)));
        if ct-nav(j)>gp
            % check for feasibility
            [e,c]=find(s==tbj(1));
e=e(c~=j);
c=c(c~=j);
flag=0;
gp3=0;
end
end
end
end
end
end
for d=1:length(e)
    if sf(e(d),c(d))>ct && sf(e(d),c(d))-t(tbj(1),c(d))<ct+t(tbj(1),j)
        gp3=max(gp3,sf(e(d),c(d))-ct);
        flag=length(e);
    end
    while flag>0
        for d=1:length(e)
            if sf(e(d),c(d))>ct+gp3 && sf(e(d),c(d))-t(tbj(1),c(d))<ct+t(tbj(1),j)+gp3
                gp3=sf(e(d),c(d))-ct;
                flag=length(e);
            else
                flag=flag-1;
            end
        end
    end
    gap(1)=gp3;
end
else
    % Check feasibility of tbj(b)
    [e,c]=find(s==tbj(b));
    e=e(c~=j);
    c=c(c~=j);
    flag=0;
    gp2=0;
    for d=1:length(e)
        if sf(e(d),c(d))>ct && sf(e(d),c(d))-t(tbj(b),c(d))<ct+t(tbj(b),j)
            gp2=max(gp2,sf(e(d),c(d))-ct);
            flag=length(e);
        end
    end
    while flag>0
        for d=1:length(e)
            if sf(e(d),c(d))>ct+gp2 && sf(e(d),c(d))-t(tbj(b),c(d))<ct+t(tbj(b),j)+gp2
                gp2=sf(e(d),c(d))-ct;
                flag=length(e);
            else
                flag=flag-1;
            end
        end
    end
    gap(b)=gp2;
end
%select ctnr to be scheduled at ct
d = find(gap == min(gap), 1);
bunch = [bunch tbj(d)];
dg = [dg gap(d)];
ct = ct + gap(d) + t(tbj(d), j);
tbj = tbj(tbj == tbj(d));
end
end

% Schedule bunch (with gaps if necessary, dg)
for b = 1:length(bunch)
    nav(j) = nav(j) + t(bunch(b), j) + dg(b);
sf(ic(j), j) = nav(j);
ft(bunch(b)) = max(ft(bunch(b)), nav(j));
temp = s(ic(j):end, j);
s(ic(j):end, j) = [bunch(b); temp(temp ~= bunch(b))];
ic(j) = ic(j) + 1;
tbs(:, j) = [tbs(tbs(:, j) ~= bunch(b), j); 0];
end
if tbs(1, j) > 0
    nt(j) = t(tbs(1, j), j);
end
end
end

d = find(gap == min(gap), 1);
bunch = [bunch tbj(d)];
dg = [dg gap(d)];
ct = ct + gap(d) + t(tbj(d), j);
tbj = tbj(tbj == tbj(d));
end
end

% Calculate tardiness(i) and total
trd = max(0, ft - due);
if sum(trd) < Tmin
    Tmin = sum(trd);
    Mtrd = max(trd);
sopt = s;
sfopt = sf;
if Tmin == LB
    return
end
elseif sum(trd) == Tmin && max(trd) < Mtrd
    Tmin = sum(trd);
    Mtrd = max(trd);
sopt = s;
sfopt = sf;
end
end
Appendix F

MATLAB functions for complete enumeration search and scheduling procedures for special case problems

function [RR,T,sopt,z] = k3n5CELB(Risk,PTFR,wts,due)
% File k3n5CELB.m
% Runs calculations for 3 stations, 5 containers
% Full enumeration with LoB to limit schedule iterations
% returns the residual risk (RR), tardiness (T), schedule (s)
% input: weight of tardiness (wT) and due times (due)
% output: RR, T, schedule, z=Rs*(1-wT)+Ts*wT

n=size(Risk,1);
k=size(Risk,2);
m=size(wts,2);
y=zeros(size(Risk));
z=2*ones(1,m);
yr=zeros(size(Risk));
yr(1,:)=[0,0,0];
yr(2,:)=[1,0,0];
yr(3,:)=[1,1,0];
yr(4,:)=[1,0,1];
yr(5,:)=[0,1,0];
yr(6,:)=[0,1,1];
yr(7,:)=[0,0,1];
yr(8,:)=[1,1,1];

file2=fopen('C:\Program Files\MATLAB\R2009b\work\CELB5scratch.txt','at');
fprintf(file2,'n scratch output from k3n5CELB');
fprintf(file2,'n a, b, prev: sum(sum(y)), toc');
[Tmx, Rmx, Rmn]=Scaling(Risk, PTFR, due);
tic;

for a=1:8
    for b=1:8
        fprintf(file2,'n %i %i %i %f',a, b, sum(sum(y)), toc);
        for c=1:8
            for d=1:8
                for e=1:8
                    y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:)];
                    %subroutine 'rescalc' returns residual risk PRes
                end
            end
        end
    end
end

for a=1:8
    for b=1:8
        fprintf(file2,'n %i %i %i %f',a, b, sum(sum(y)), toc);
        for c=1:8
            for d=1:8
                for e=1:8
                    y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:)];
                    %subroutine 'rescalc' returns residual risk PRes
                end
            end
        end
    end
end
PRes = rescalc(Risk, PTFR, y);
RRo = sum(sum(PRes));
TminLB = LoB(y, due);
for mi = 1:m
    zm(mi) = (RRo - Rmn) * (1 - wts(mi)) / (Rmx - Rmn) + wts(mi) * TminLB / Tmx;
end

if sum(sum(y)) == 0
    Ta = 0;
s = zeros(n + k, k);
elseif zm >= z
    Ta = TminLB;
else
    % subroutine 'bestsched' returns min tardiness
    [Ta, s] = bestsched(y, due, TminLB);
end
for mi = 1:m
    if ((RRo - Rmn) * (1 - wts(mi)) / (Rmx - Rmn) + wts(mi) * Ta / Tmx) < z(m)
        z(m) = (RRo - Rmn) * (1 - wts(mi)) / (Rmx - Rmn) + wts(mi) * Ta / Tmx;
        RR(mi) = RRo;
        T(mi) = Ta;
        yo = y;
        so = s;
    end
end
end
end
end
fclose(file2);

function [Tmin, sopt] = bestsched(y, due, LB)
% File bestsched.m
% Formatted for 3 stations
% Builds and compares all possible schedules to find the one that
% returns the minimum total tardiness (deemed the best schedule)
% input: allocation(y), due times(due), lower bound(LB)
% output: total tardiness(Tmin) and start time matrix(sopt)

n = size(y, 1);
k=size(y,2);
Tmin=10000;
Mtrd=10000;
trd=zeros(size(due));
s=zeros(n+k,k);
sopt=s;
if max(sum(y))>10 || sum(sum(y))==0 || k~=3
    return
end
[ctnr,stn]=find(y);

% Operation x is for container:ctnr(x) and station:stn(x)

% Operations by each station:
ops1=find(stn==1);
ops2=find(stn==2);
ops3=find(stn==3);
prm1=perms(ops1);
prm2=perms(ops2);
prm3=perms(ops3);

for f=1:size(prm1,1)
    for g=1:size(prm2,1)
        for h=1:size(prm3,1)
            oporder=[prm1(f,:),prm2(g,:),prm3(h,:)];
            % given order of operations for scheduling
            s=zeros(n+k,k);
            trd=zeros(size(due));
            for b=1:size(oporder,2)
                % current operation to be scheduled
                c=1;
                scheduled=0;
                while scheduled==0 && c<=size(s,1)
                    if s(c,stn(oporder(b)))==0 &
                        isempty(find(s(c,:)==ctnr(oporder(b)), 1))
                        % stn is available and ctnr is not scheduled
                        s(c,stn(oporder(b)))=ctnr(oporder(b));
                        scheduled=1;
                    else
                        c=c+1;
                    end
                end
            end
            % Calculate tardiness(i) and total
            for b=1:n
                [d,e]=find(s==b);
if isempty(d)
    trd(b)=0;
else
    trd(b)=max(0,max(d)-due(b));
end
end

if sum(trd)<Tmin
    Tmin=sum(trd);
    Mtrd=max(trd);
    sopt=s;
    if Tmin==LB
        return
    end
endif
elseif sum(trd)==Tmin && max(trd)<Mtrd
    Tmin=sum(trd);
    Mtrd=max(trd);
    sopt=s;
end
end
end
Appendix G

MATLAB functions for complete enumeration search and scheduling procedures for general case problems

function [RR,T,sopt,sfo,z] = k3CEwt(Risk,PTFR,wts,due,t)
% File k3CEwt.m
% Runs calculations for 3 stations, n=5:10 containers
% Full enumeration with LBt to limit schedule iterations
% returns the residual risk (RR), tardiness (T),
% (n,k) matrices of container schedule and finish times
% input: weight of tardiness (wT) and due times (due)
% output: RR, T, schedule, z=Rs*(1-wT)+Ts*wT

n=size(Risk,1);
k=size(Risk,2);
m=size(wts,2);
y=zeros(size(Risk));
z=2*ones(1,m);
yr=zeros(size(Risk));
yr(1,:)=[0,0,0];
yr(2,:)=[1,0,0];
yr(3,:)=[1,1,0];
yr(4,:)=[1,0,1];
yr(5,:)=[0,1,0];
yr(6,:)=[0,1,1];
yr(7,:)=[0,0,1];
yr(8,:)=[1,1,1];

file2=fopen(['C:\Program Files\MATLAB\R2009b\work\k3n' int2str(n) '\Scratch.txt'],'at');
fprintf(file2,['\n scratch output from k3CEwt']);
fprintf(file2,'\n a, b, prev: sum(sum(y)), toc');
[Tmx, Rmx, Rmn]=Scalingwt(Risk, PTFR, due, t);
tic;

if n==5
    for a=1:8
        for b=1:8
            fprintf(file2,'\n %i %i %i %f',a, b, sum(sum(y)), toc);
            for c=1:8
                for d=1:8
                    for e=1:8

y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:)];
%subroutine 'rescalc' returns residual risk PRes
PRes=rescalc(Risk,PTFR,y);
RRo=sum(sum(PRes));
TminLB=LBt(y, due, t);
zm=(RRo-Rmn)*(1-wts)/(Rmx-Rmn)+wts*TminLB/Tmx;
if sum(sum(y))==0
    Ta=0;
    s=zeros(n,k);
    sf=s;
else if zm>=z
    Ta=TminLB;
else
    %subroutine 'bestschedwt' returns min tardiness
    [Ta,s,sf]=bestschedwt(y, due, t, TminLB);
end
for mi=1:m
    if ((RRo-Rmn)/(Rmx-Rmn)+wts(mi)*Ta/Tmx)<z(mi)
        z(mi)=(RRo-Rmn)/(Rmx-Rmn)+wts(mi)*Ta/Tmx;
        RR(mi)=RRo;
        T(mi)=Ta;
        yopt(:,:,mi)=y;
        sopt(:,:,mi)=s;
        sfo(:,:,mi)=sf;
        end
end
end
end
end
end

elseif n==6
for a=1:8
    for b=1:8
        fprintf(file2,'\n %i %i %i %f',a, b, sum(sum(y)), toc);
        for c=1:8
            for d=1:8
                for e=1:8
                    for f=1:8
                        y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:);yr(f,:)];
                        %subroutine 'rescalc' returns residual risk PRes
                        PRes=rescalc(Risk,PTFR,y);
                        RRo=sum(sum(PRes));
                        TminLB=LBt(y, due, t);
                        end
                end
            end
        end
    end

end
end
end
end
end
end
end
zm=(RRo-Rmn)/(Rmx-Rmn)+wts*TminLB/Tmx;
if sum(sum(y))==0
    Ta=0;
    s=zeros(n,k);
    sf=s;
elseif zm>=z
    Ta=TminLB;
else
    %subroutine 'bestschedwt' returns min tardiness
    [Ta,s,sf]=bestschedwt(y,due,t,TminLB);
end
for mi=1:m
    if ((RRo-Rmn)/(Rmx-Rmn)+wts(mi)*Ta/Tmx)<z(mi)
        z(mi)=(RRo-Rmn)/(Rmx-Rmn)+wts(mi)*Ta/Tmx;
        RR(mi)=RRo;
        T(mi)=Ta;
        yopt(:,:,mi)=y;
        sopt(:,:,mi)=s;
        sfo(:,:,mi)=sf;
    end
end
end
end
end
end
end
elseif n==7
    for a=1:8
        for b=1:8
            fprintf(file2,'\n %i %i %i %f',a, b, sum(sum(y)), toc);
            for c=1:8
                for d=1:8
                    for e=1:8
                        for f=1:8
                            for g=1:8
                                y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:);yr(f,:);yr(g,:)];
                                %subroutine 'rescalc' returns residual risk PRes
                                PRes=rescalc(Risk,PTFR,y);
                                RR0=sum(sum(PRes));
                                TminL=LBt(y,due,t);
                                zm=(RR0-Rmn)/(Rmx-Rmn)+wts*TminLB/Tmx;
                                if sum(sum(y))==0
                                    Ta=0;
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
s=zeros(n,k);
sf=s;
elseif zm>=z
    Ta=TminLB;
    elseif
        subroutine 'bestschedwt' returns min tardiness
        [Ta,s,sf]=bestschedwt(y,due,t,TminLB);
    end
    for mi=1:m
        if ((RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx)<z(mi)
            z(mi)=(RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx;
            RR(mi)=RRo;
            T(mi)=Ta;
            yopt(:,mi)=y;
            sopt(:,mi)=s;
            sfo(:,mi)=sf;
        end
    end
elseif n==8
    for a=1:8
        for b=1:8
            fprintf(file2,’\n %i %i %i %f’,a, b, sum(sum(y)), toc);
        end
    end
    for c=1:8
        for d=1:8
            for e=1:8
                for f=1:8
                    for g=1:8
                        for h=1:8
                            y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:);yr(f,:);yr(g,:);yr(h,:)];
                            subroutine 'rescalc' returns residual risk PRes
                            PRes=rescalc(Risk,PTFR,y);
                            RRo=sum(sum(PRes));
                            TminLB=LBt(y,due,t);
                            zm=(RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts*Ta/Tmx;
                            if sum(sum(y))==0
                                Ta=0;
                                s=zeros(n,k);
elseif zm>=z
    Ta=TminLB;
else
    % subroutine 'bestschedwt' returns min tardiness
    [Ta,s,f]=bestschedwt(y,due,t,TminLB);
end
for mi=1:m
    if ((RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx)<z(mi)
        z(mi)=(RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx;
        RR(mi)=RRo;
        T(mi)=Ta;
        yopt(:,:,mi)=y;
        sopt(:,:,mi)=s;
        sfo(:,:,mi)=f;
    end
end
elseif n==9
    for a=1:8
        for b=1:8
            fprintf(file2,'n %i %i %i %f',a, b, sum(sum(y)), toc);
        end
        for c=1:8
            for d=1:8
                for e=1:8
                    for f=1:8
                        for g=1:8
                            for h=1:8
                                for i=1:8
                                    y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:);yr(f,:);yr(g,:);yr(h,:);yr(i,:)];
                                    % subroutine 'rescalc' returns residual risk PRes
                                    PRes=rescalc(Risk,PTFR,y);
                                    RRo=sum(sum(PRes));
                                    TminLB=LBt(y,due,t);
                                    zm=(RRo-Rmn)*(1-wts)/(Rmx-Rmn)+wts*TminLB/Tmx;
                                    if sum(sum(y))==0
                                        Ta=0;
                                    end
    end
end
end
end
end
end
end
end
end
end
s=zeros(n,k);
sf=s;
elseif zm>=z
 Ta=TminLB;
else
 % subroutine 'bestschedw' returns min tardiness
 [Ta,s,sf]=bestschedw(y,due,t,TminLB);
end
for mi=1:m
 if ((RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx)<z(mi)
  z(mi)=(RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx;
  RR(mi)=RRo;
  T(mi)=Ta;
  yopt(:,:,mi)=y;
  sopt(:,:,mi)=s;
  sfo(:,:,mi)=sf;
 end
end
end
end
end
end
end
end
end
endif n==10
 for a=1:8
  for b=1:8
   fprintf(file2,\n %i %i %i %f',a, b, sum(sum(y)), toc);
   for c=1:8
    for d=1:8
     for e=1:8
      for f=1:8
       for g=1:8
        for h=1:8
         for i=1:8
          for j=1:8
           y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:);yr(f,:);yr(g,:);yr(h,:);yr(i,:);yr(j,:)];
           % subroutine 'rescalc' returns residual risk PRes
           PRes=rescalc(Risk,PTFR,y);
           RRo=sum(sum(PRes));
           TminLB=LBt(y,due,t);
           elseif n==10
            for a=1:8
             for b=1:8
              fprintf(file2,\n %i %i %i %f',a, b, sum(sum(y)), toc);
              for c=1:8
               for d=1:8
                for e=1:8
                 for f=1:8
                  for g=1:8
                   for h=1:8
                    for i=1:8
                     for j=1:8
                      y=[yr(a,:);yr(b,:);yr(c,:);yr(d,:);yr(e,:);yr(f,:);yr(g,:);yr(h,:);yr(i,:);yr(j,:)];
                      % subroutine 'rescalc' returns residual risk PRes
                      PRes=rescalc(Risk,PTFR,y);
                      RRo=sum(sum(PRes));
                      TminLB=LBt(y,due,t);
zm=(RRo-Rmn)*(1-wts)/(Rmx-Rmn)+wts*TminLB/Tmx;
if sum(sum(y)) == 0
    Ta=0;
    s=zeros(n,k);
    sf=s;
elseif zm >= z
    Ta=TminLB;
else
    % subroutine 'bestschedwt' returns min tardiness
    [Ta,s,sf]=bestschedwt(y,due,t,TminLB);
end
for mi=1:m
    if ((RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx)<z(mi)
        z(mi)=(RRo-Rmn)*(1-wts(mi))/(Rmx-Rmn)+wts(mi)*Ta/Tmx;
        RR(mi)=RRo;
        T(mi)=Ta;
        yopt(:,:,mi)=y;
        sopt(:,:,mi)=s;
        sfo(:,:,mi)=sf;
    end
end

function [Tmin,sopt,sfo] = bestschedwt(y,due,t,LB)
% File bestschedwt.m
% Formatted for 3 stations
% Builds and compares all possible schedules to find the one that
% returns the minimum total tardiness (deemed the best schedule)
% input: allocation(y), due times, processing times, LB
% output: total tardiness(Tmin) and
% (n,k) matrices of container schedule and finish times
n=size(y,1);
k=size(y,2);
LrgT=10000;
Tmin=LrgT;
Mtrd=LrgT;
s=zeros(n,k);
sopt=s;
sfo=s;

if max(sum(y))>10 || sum(sum(y))==0 || k~=3
    return
end
[~,ind]=sort(sum(y.*t),'descend');
% stations in order of total processing time
[ctnr,sn]=find(y);
% container: ctnr(x) and station: stn(x)
f=ind(1);
g=ind(2);
h=ind(3);

% All operations in f, ctnr(stn==f)
prmf=perms(ctnr(stn==f));
prmg=perms(ctnr(stn==g));
prmh=perms(ctnr(stn==h));
% Schedule all operations in f, g, h
for pf=1:size(prmf,1)
    for pg=1:size(prmg,1)
        for ph=1:size(prmh,1)
            s(:,f)=[prmf(pf,:); zeros(1,n-size(prmf,2))];
            s(:,g)=[prmg(pg,:); zeros(1,n-size(prmg,2))];
            s(:,h)=[prmh(ph,:); zeros(1,n-size(prmh,2))];
            trd=zeros(n,1);
            ft=trd;
            tbs=s;
            sf=zeros(size(s));
            % Calculate feasible finish times for ctnrs
            % Schedule f continuously, then g, h
            % avoid conflict by delaying start
            nav=zeros(1,k);
            % next available time for f, g, h
            nt(f)=t(max(1,s(1,f)),f); % next tij for f, g, h
            nt(g)=t(max(1,s(1,g)),g);
            nt(h)=t(max(1,s(1,h)),h);
            ic=ones(1,k);
            while sum(sum(tbs))>0
if tbs(1,f)>0
  % Schedule tbs(1,f) at f
  nav(f)=nav(f)+nt(f);
  sf(ic(f),f)=nav(f);
  ft(tbs(1,f))=max(ft(tbs(1,f)),nav(f));
  tbs(:,f)=[tbs(2:end,f);0];
  ic(f)=ic(f)+1;
  if tbs(1,f)>0
    nt(f)=t(tbs(1,f),f);
  end
elseif tbs(1,g)>0
  % check feasibility and then schedule g
  % find start time (st) such that [st, st+nt) doesn't
  % overlap with [sf-t,sf] of same ctnr at f (only)
  st=nav(g);
  cv=find(s(:,f)==tbs(1,g),1);
  % [st,st+nt] vs. [sf-t, sf]
  if sf(cv,f)>st & sf(cv,f)-t(tbs(1,g),f)<st+nt(g)
    st=sf(cv,f);
  end
  % Schedule tbs(1,g) at g
  nav(g)=st+nt(g);
  sf(ic(g),g)=nav(g);
  ft(tbs(1,g))=max(ft(tbs(1,g)),nav(g));
  tbs(:,g)=[tbs(2:end,g);0];
  ic(g)=ic(g)+1;
  if tbs(1,g)>0
    nt(g)=t(tbs(1,g),g);
  end
elseif tbs(1,h)>0
  % schedule h where feasible
  st=nav(h);
  % look for conflict with ind(c), c=1:2, and repeat until none
  stc=1;
  while stc>0
    % repeats until no conflict
    stc=2;
    for c=1:2  % (k-1)
      cv=find(s(:,ind(c))==tbs(1,h),1);
      if sf(cv,ind(c))>st & sf(cv,ind(c))-t(tbs(1,h),ind(c))<st+nt(h)
        st=sf(cv,ind(c));
      else
        stc=stc-1;
      end
    end
end
% Schedule tbs(1,h) at h
nav(h)=st+nt(h);
sf(ic(h),h)=nav(h);
ft(tbs(1,h))=max(ft(tbs(1,h)),nav(h));
tbs(:,h)=[tbs(2:end,h);0];
ic(h)=ic(h)+1;
if tbs(1,h)>0
    nt(h)=t(tbs(1,h),h);
end
end
end

% Calculate tardiness(i) and total
trd=max(0,ft-due);
if sum(trd)<Tmin
    Tmin=sum(trd);
    Mtrd=max(trd);
    sopt=s;
    sfo=sf;
    if Tmin==LB
        return
    end
elseif sum(trd)==Tmin & max(trd)<Mtrd
    Tmin=sum(trd);
    Mtrd=max(trd);
    sopt=s;
    sfo=sf;
end
end
end
end
Curriculum Vitae

Christina M. Young

Education

Rutgers, The State University of New Jersey
Dates attended: September 1999- October 2010

October 2010 Ph.D. in Industrial and Systems Engineering
October 2005 M.S. in Industrial and Systems Engineering
May 2005 M.S. in Statistics
May 2003 B.S. in Industrial Engineering, Highest Honors

Publications


