DESIGNING AVIATION SECURITY SYSTEMS: THEORY AND PRACTICE

BY

LAURA A. MCLAY

B.S., University of Illinois at Urbana-Champaign, 2000
M.S., University of Illinois at Urbana-Champaign, 2001

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical and Industrial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

Abstract

Designing effective aviation security systems has become a problem of national concern. Since September 11, 2001, numerous changes have been made to aviation security systems. Many of these changes have been politically driven, rather than driven by coordinated, systematic analysis and planning, and as a result, the analysis of how aviation security systems operate continues to lag well behind their actual implementation.

This dissertation applies operations research methodologies to aviation passenger screening problems, an important and highly visible aspect of aviation security. Effective passenger screening systems optimally allocate and use security assets and technologies to prevent terrorist attacks. Designing effective passenger screening systems becomes more critical as new screening technologies are made available and as security intelligence improves. Two basic passenger screening models are identified, and algorithms for obtaining optimal and near-optimal solutions for these problems are reported. One of the passenger screening models is extended to handle dynamic passenger arrivals. Exact and approximation algorithms are developed for three knapsack problem variations whose structures are similar to those of the passenger screening models. Finally, a cost-benefit analysis is performed to quantify the tradeoffs between having effective, expensive screening technologies and accurate security intelligence. The key implication obtained is that investing in new, expensive technologies for preventing successful attacks is warranted only if security intelligence is accurate.

Next-generation aviation security systems need not merely be makeshift political solutions for mending complex problems; they can be the result of modeling, analysis, and planning. This dissertation provides a systematic approach for designing and analyzing aviation security systems, which provides insight into the operation of passenger screening systems and guidance for the design of next-generation aviation security systems.

For Courtlan and Eleanor

Acknowledgments

This project would not have been possible without the support of many people. I especially want to thank

- my adviser, Dr. Sheldon H. Jacobson for teaching me how to become a successful student and academic, giving me the freedom to explore many problems during the time we have worked together, giving me excellent advice when I needed it, and having confidence in my abilities.
- Dr. John E. Kobza, for giving me excellent guidance and feedback on my research.
- my committee members, Dr. Udatta S. Palekar, Dr. Dieter Vandenbussche, and Dr. Uday V. Shanbhag for their guidance and feedback.
- Dr. Diego Klabjan and Dr. Deborah L. Thurston for their feedback on my dissertation proposal.
- Dr. David E. Goldberg for his guidance on my Masters thesis and encouraging me to pursue a Ph.D.
- the Department of Mechanical and Industrial Engineering for awarding me the Alumni Board Teaching Fellowship, which provided me with the opportunity to gain teaching experience, and for awarding me the George W. Harper Award.
- the Arms, Control, Disarmament, and International Security (ACDIS) program for awarding me the ADCIS Thesis Initiation Fellowship that provided funding for this research.
- the National Science Foundation (DMI-0114499 and DMI-0114046) and the Air Force Office of Scientific Research (FA9550-04-1-0110) that provided funding for this research.

- several people at the Transportation Security Administration within the United States Department of Homeland Security for their feedback, especially Dr. John J. Nestor, Dr. Lyle Malotky, Mr. Michael McCormick, Ms. Theresa Hasty, and Dr. Fred Roder.

- my labmates, Hemanshu Kaul, Shane N. Hall, Alex Nikolaev, Gio Kao, Ruben Proano, and Adrian Lee for their friendship, support, and tolerance for the massive amount of coffee and oatmeal I cooked in the lab while working on my dissertation.

- for "Group" and especially Carla for supporting and challenging me.

- my parents, Jim and Julie Albert, for recognizing my potential, encouraging me to pursue postgraduate education, encouraging me to hurdle every obstacle I encountered, and being my biggest cheerleaders.

- my sister and brother-in-law, Jill and Tim Sullivan, my brother, Brian Albert, my grandparents, Helena Albert and Frank Trunk, and many friends, especially Yung-Tae Kim, who endured this long process with me, always offering support.

- my husband, Court, for his patience, support, and love throughout the graduate school experience, and our daughter, Eleanor, for all of the little joys she has provided since she entered this world.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Motivation

The terrorist events on September 11, 2001 will forever alter the way in which aviation security is viewed. Over the past five years, there have been numerous changes to all aspects of aviation security systems, all of which have been designed to prevent a reoccurrence of the events on September 11, 2001. Many of the changes implemented have been politically driven. For example, several billion dollars were invested in security devices following September 11, 2001 before any type of systematic analysis of aviation security systems was performed [55]. Coordinated analysis and planning has the potential to determine how taxpayer dollars can be optimally spent and how security system assets can be optimally used.

Next-generation aviation security systems need not merely be makeshift political solutions for mending complex problems; they can be the result of modeling, analysis, and planning. This dissertation provides a systematic approach for designing and analyzing passenger screening systems, an important and highly visible aspect of aviation security. The models and analysis provide insight into the operation of passenger screening systems and guidance for the design of next-generation aviation security systems.

There are two basic approaches to aviation security screening: uniform screening and selective screening. Uniform screening subjects every passenger and their baggage to identical security screening procedures. The argument for uniform screening is that anyone could pose a risk, and hence, all passengers should be screened using the most effective technology and procedures available. The 100% baggage screening mandate, which requires all checked baggage to be screened by a federally certified explosive detection technology (effective December 31, 2002), is a move towards uniform screening [54]. One disadvantage of uniform screening is that it can be very costly to apply expensive new technologies to every passenger or bag. Butler and Poole [12] and Poole and Passantino [67] argue that 100% checked baggage screening is not cost-effective, and suggest that

creating multiple levels of security for screening passengers may be more effective than treating all passengers the same.

Selective screening, the alternative to uniform screening, selectively applies security technologies and procedures to a subset of passengers. The argument for selective screening is that most passengers do not pose a risk, and hence, expensive security technologies need not be used on all passengers. Selective screening subjects passengers perceived as high-risk to closer scrutiny by screening them and their baggage with more sensitive and accurate technologies and procedures, while passengers perceived as low-risk are subjected to lower levels of scrutiny. This approach requires that a prescreening system perform a risk assessment of each passenger prior to the passenger's arrival. A weakness of selective screening is that it can assign an incorrect degree of risk to a passenger, either by error or through "gaming" of the system by a terrorist.

1.1 Domestic Aviation Security History

Hijacking attempts were a serious breech of early aviation security in the United States. They were relatively infrequent until 1968 when twenty hijacking attempts occurred on US aircraft and fifteen hijacking attempts occurred on foreign aircraft [19]. On September 11, 1970, then President Nixon announced a program of deploying surveillance equipment to the nation's airports to reduce the increased numbers of hijacking attempts. Furthermore, air carriers worked with the Departments of Defense and Transportation to determine whether x-ray devices and metal detectors could be integrated into airports to screen passengers and their carry-on baggage. On February 1, 1972, the Federal Airline Administration (FAA) announced that all passengers were to be screened by at least one approved method, which included a behavioral profile, metal detector, identification check, and physical search. When hijacking attempts persisted, the FAA adopted emergency rules on December 5, 1972, requiring air carriers to use screening procedures to prevent passengers from bringing weapons and explosives onto the aircraft [57]. Figure 1.1 shows the number of hijacking attempts from 1947 to 2003 [19]. There were at least twenty-three total domestic hijacking attempts during the years from 1969 to 1972, but the number of hijacking attempts plummeted to two in 1973. There were relatively few hijacking attempts after 1972, with six or fewer domestic hijacking attempts in any single year since 1984.

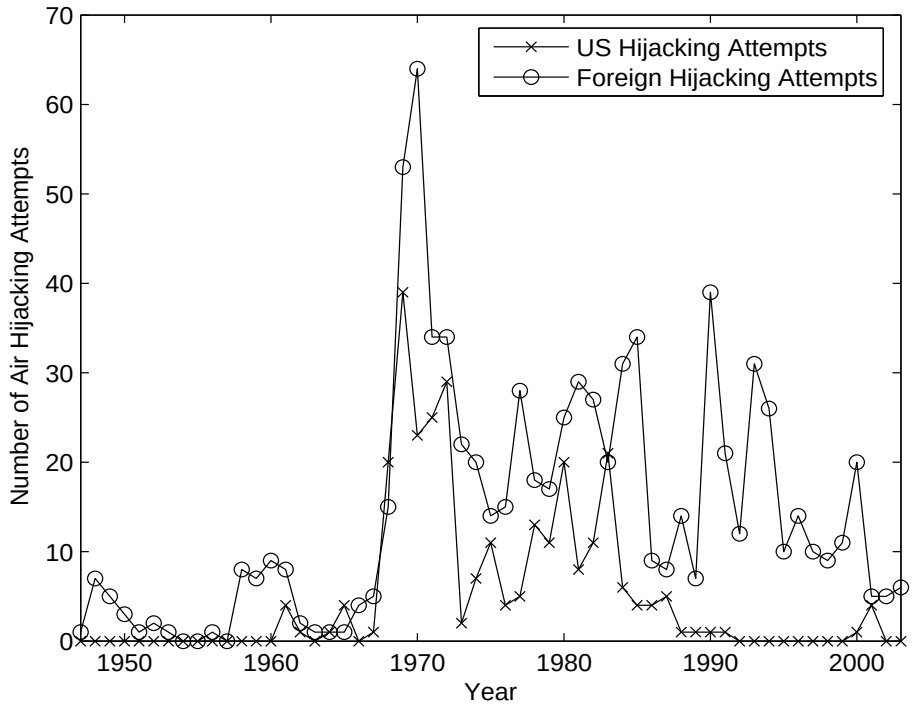

Figure 1.1: Foreign and domestic air hijacking attempts, 1947 – 2003

Aviation security operations at the nation's airports did not significantly change from December 1972 to 1996. However, the destruction of Pan Am Flight 103 over Lockerbie, Scotland on December 21, 1988 and the crash of TWA Flight 800 on July 17, 1996 led to the creation of the Commission on Aviation Safety and Security on July 25, 1996, headed by then Vice-President Al Gore. The Commission on Aviation Safety and Security recommended that the aviation industry improve security using existing explosive detection technologies, automated passenger prescreening, and positive passenger-baggage matching (PPBM). Moreover, the Federal Aviation Administration (FAA) had been working with the airlines to annually purchase and deploy explosive detection systems (EDSs) at airports throughout the United States. From 1998 until September 11, 2001, EDSs were only used to screen checked baggage of *selectee* passengers, those who were not cleared by the Computer-Aided Passenger Prescreening System (CAPPS), a computer risk assessment system developed in conjunction with the FAA, Northwest Airlines, and the United States Department of Justice [79]. The checked baggage of *nonselectee* passengers, those who were cleared by such a system, received no additional security attention. There were no further differences between selectee

and nonselectee passengers. During this time, approximately 95% of passengers were classified as nonselectees [2].

The terrorist events on September 11, 2001 prompted Congress to enact the Aviation and Transportation Security Act on November 19, 2001, which transferred aviation security from the FAA to the newly-created Transportation Security Administration (TSA), then part of the Department of Transportation (now part of the Department of Homeland Security). This act prescribed additional security procedures to be implemented [54, 55], including screening all checked baggage for explosives by EDSs or alternative techniques. Given such a policy, there would no longer be any distinction between selectee and nonselectee passengers, since all checked baggage would be screened. In order to meet this requirement, 1100 EDSs and 5000 explosive trace detection systems (ETDs) were deployed to the nation's 429 commercial airports [55]. Before these security devices were deployed, some checked baggage was physically screened for explosives by hand searches, and PPBM removed checked baggage from aircraft if the passengers who owned the baggage did not board the aircraft. The Aviation and Transportation Security Act also required that security personnel become federal employees (this was accomplished on November 18, 2002), as well as improved perimeter security and cargo screening.

A number of security experts suggest that greater scrutiny of passengers perceived as greater risks (from a security standpoint) is a more effective approach to aviation security than treating all passengers the same. Butler and Poole [12] suggest that the TSA's policy of 100% checked baggage screening is not cost-effective and that enhancing CAPPS, a binary screening paradigm, to a multilevel screening system would be a more effective approach to process airline passengers. Poole and Passantino [67] endorse *risk-based airport security*, partitioning passenger and baggage security devices in proportion to the level of perceived risk. They suggest that multiple levels of security for processing passengers may be more effective than treating all passengers as indistinguishable (from a security standpoint).

The TSA responded to these proposals through the development of CAPPS II, an enhanced computer-based system for systematically prescreening passengers. CAPPS II was a revision of CAPPS, which did not classify all of the 19 terrorists on September 11, 2001 as selectees. Although the true number of these terrorists classified as selectees is classified, it has been estimated in the

public domain that between 6 and 11 of the terrorists were classified as selectees [3, 56, 71].

On July 14, 2004, the TSA announced that CAPPS II had been dismantled over privacy concerns despite having invested $100M into its development. Shortly thereafter, the TSA announced plans to replace CAPPS II with *Secure Flight*, an automated system akin to CAPPS II (i.e., a system designed to partition passengers into risk classes) for passenger prescreening [85]. Secure Flight partitions passengers into three risk classes: nonselectees, selectees, and a third class of passengers not permitted to fly based on federal terrorist watch lists. However, the number of passengers in the third group is anticipated to be extremely small. Secure Flight distinguishes selectees and nonselectees by requiring additional screening (such as hand searches) for selectees and their carry-on baggage. How Secure Flight operates is considered highly sensitive and may change based on changes in national or international situations, intelligence information, or the risk level of the Homeland Security Advisory System [78].

Several other changes have been made to aviation security since September 11, 2001. The set of items prohibited from being carried onto aircraft have changed several times. Soon after September 11, 2001, box cutters, knives, nail clippers, and lighters were prohibited. Most recently, in December 2005, the TSA announced that small knives, scissors, and tools would be permitted to be carried on to commercial flights [82]. Moreover, the TSA has been working toward implementing a Registered Traveler Program that will reduce the amount of security scrutiny for registered travelers who provide personal and biometric information in order to verify their identities [83]. A five-airport pilot program is being used to determine the role of the Registered Traveler Program and its impact on passenger screening systems [78]. The TSA has also been deploying explosive detection trace portals and Explosives Detection Canine Teams, as they become available [86, 81]. The Federal Air Marshal Service (FAMS) program began in 1968, when it was called the Sky Marshal Program. FAMS was expanded after September 11, 2001, and the number of federal air marshals incerased from fewer than fifty to several thousand during this time [84].

The primary objective of all these efforts is to improve security operations at the nation's commercial airports while reducing the time required to perform passenger screening. To meet these objectives, the TSA must develop new security system paradigms that can optimally use and simultaneously coordinate several security technologies and procedures. New security procedures

put in place by the TSA have the potential to affect a large number of passengers and baggage. Note that in 2005, there were over 700 million passengers, with forecasts of over one billion passengers by 2015 [77].

1.2 Literature Review

Several integer programming and discrete optimization models have been formulated to describe aviation security problems [49]. Jacobson et al. [27] provide a framework for measuring the effectiveness of a baggage screening security device deployment at a particular station. A station is a set of airport facilities that share security resources, and there may be several stations in a large, hub airport. Jacobson et al. [27, 32] introduce three performance measures for baggage screening security systems and introduce models to assess the security effect for single or multiple stations. The models developed consider using EDSs to screen selectee checked baggage and to possibly screen a proportion of nonselectee checked baggage. The three performance measures considered are to maximize the utilization of baggage screening devices, to maximize the number of flights that are covered (i.e., all selectee baggage on the covered flights have been screened), and to maximize the number of passengers on the flights that are covered.

Jacobson et al. [30] formulate problems that model multiple sets of flights originating from multiple stations subject to a finite amount of security resources. Examples illustrate strategies that may provide more robust device allocations across all performance measures. Jacobson et al. [31] and Virta et al. [88] consider the impact of originating and transferring passengers on the effectiveness of baggage screening security systems. In particular, they consider classifying selectees into two types; those at their point of origin and those transferring. They construct integer programming models for problems that consider multiple sets of flights originating from multiple airports. This is noteworthy since at least two of the hijackers on September 11, 2001 were transferring passengers.

Babu et al. [1] investigate the possible benefit from using multiple risk groups for screening passengers using linear programming models. The objective of their model is to minimize the probability of the system giving a false alarm, subject to false clear and screening time constraints. They find that using multiple risk groups are beneficial for security, even when a prescreening

system is not used to differentiate passenger risk.

Several papers perform a cost-benefit analysis of using checked baggage screening devices. They study the tradeoffs between screening only selectee baggage and screening both selectee and nonselectee baggage for explosives. Virta et al. [89] conclude that the marginal increase in security per security dollar spent is significantly lower when both selectee and nonselectee baggage is screened than when only selectee baggage is screened. Jacobson et al. [28] extend this model to consider the indirect effect of deterrence on the level of threat at an airport. McLay et al. [50] examine checked baggage screening systems that use a prescreening system and different baggage screening devices, one to screen selectee baggage and the other to screen nonselectee baggage. They conclude that using expensive and accurate baggage screening technologies on selectees is warranted only if there is an effective prescreening system in place.

Several papers analyze the tradeoffs between the false alarm and false clear rates associated with passenger screening systems. Kobza and Jacobson [38, 39] illustrate how the false alarm and false clear rates can be interpreted as Type I and Type II errors and analyze their relationship for screening systems consisting of multiple devices. Jacobson et al. [29] introduce the concept of a system response function, which minimizes the false alarm rate while meeting a standard for the false clear rate by using a knapsack problem formulation.

Other research has focused on the experimental and statistical analysis of risk and security procedures on aircraft. Barnett et al. [6] reports the results of a large-scale two-week experiment at several commercial airports to test which costs and disruptions would arise from using PPBM for all flights. Barnett et al. [4] and Barnett and Higgins [5] study mortality rates on passenger aircraft and perform a statistical analysis on this data.

A frequently mentioned criticism of any system designed to classify passengers into risk classes, including CAPPS II and Secure Flight, is that such systems can be gamed through extensive trial and error sampling by a variety of passengers through the system. Carnival Booth is an algorithm that shows how threat passengers can determine a set of circumstances under which they are classified as nonselectees [17]. It shows that a system using prescreening may be less secure than systems that employ random searches. Carnival Booth takes advantage of passengers being aware of whether they are classified as selectees or nonselectees; a system using prescreening is more

difficult to defeat if passengers do not know if they are selectees or nonselectees.

Martonosi and Barnett [46] note that trial and error sampling may not increase the probability of a successful attack and that CAPPS II may not substantially improve aviation security if the screening procedures for each type of passenger are not effective. Barnett [3] suggests that CAPPS II may only improve aviation security under a particular set of circumstances and recommends that CAPPS II be transitioned from a security centerpiece to one of many components in future aviation security strategies. A number of experts agree with Barnett's assessment of CAPPS II, but are more optimistic of its potential usefulness. Caulkins [15] argues that CAPPS II may be a worthy investment since CAPPS II reduces screening costs and is an extremely small fraction of the budget allocated for passenger screening. Caulkins [15] also suggests several ways that CAPPS II could be used to increase the overall security even if terrorists are incorrectly classified as nonselectees. Cartensen [14] and Ravid [70] indicate that CAPPS II could successfully deter attacks on commercial aircraft, while Ravid [70] notes that the objective of CAPPS II is to deter terrorist events, not to capture terrorists.

Martonosi [47] addresses several passenger screening problems. One result reported is that the underlying screening process has a larger impact on reducing successful attacks than an effective prescreening system. A cost-benefit analysis of using EDSs to screen cargo, airmail, and checked baggage for explosives indicates that possible benefits depend on the type of screening policy employed. Martonosi [47] also notes that dynamically assigning security screening personnel to screening stations based on demands does not significantly reduce passenger waiting times in security lines.

Other research has modeled aviation security problems using risk analysis and game-theoretic models. Kunreuther and Heal [40] and Heal and Kunreuther [22] investigate interdependent security decisions. An example suggests that individual airlines are less likely to invest in security measures as the number of other airlines who are not investing in security measures increases.

1.3 Dissertation Overview

This dissertation introduces a systematic approach for designing and analyzing selective passenger screening systems using discrete optimization models and algorithms. It formulates problems that

model *multilevel* passenger screening strategies. Multilevel screening considers two or more levels of security to screen passengers, as opposed to the binary system in place prior to September 11, 2001. Therefore, the primary contribution of this effort is to identify models for designing multilevel screening security systems, and show how these models can be used to provide insights into the operation and performance of such systems, under the assumption that a passenger prescreening system (such as Secure Flight or CAPPS) has been implemented and is highly effective in identifying passenger risk [87].

This dissertation is organized as follows. Chapter 2 formulates the Multilevel Allocation Problem (MAP) as a discrete optimization model and integer program. Algorithms and heuristics are presented for MAP. Computational results suggest that two risk classes are sufficient for providing effective aviation security.

Chapter 3 formulates the Multilevel Passenger Screening Problem (MPSP) as a discrete optimization problem and integer program. Several theoretical properties of MPSP are reported, and these properties are illustrated on a computational example.

Chapter 4 extends MPSP to consider sequential, stochastic passenger arrivals. It is formulated as a Markov decision process and the optimal policy is found by dynamic programming. A heuristic is provided that provides a policy in real-time, and a condition is provided under which the heuristic is optimal.

Chapter 5 performs a cost-benefit analysis of checked baggage screening systems that analyzes the tradeoffs between having more accurate yet expensive screening devices and a more accurate prescreening system. The key contribution of the analysis is that the prescreening system must be accurate in assessing passenger risk in order to reduce the number of successful attacks, particularly if the proportion of passengers classified as selectees is small.

A particular case of MAP can be formulated using Knapsack Problem models and algorithms. Chapter 6 models and analyzes three knapsack variations whose structures are similar to MAP. Dynamic programming algorithms, Greedy heuristics, and fully polynomial-time approximation schemes are presented for these knapsack variations.

This dissertation is summarized in Chapter 7. The limitations of the models are made more transparent, and several directions for future research are outlined.

Chapter 2

The Multilevel Allocation Problem

This chapter formulates and analyzes a multilevel passenger screening problem [52]. The primary contribution of this effort is to identify models for designing multilevel passenger screening systems, and show how these models can be used to provide insights into the operation and performance of such systems. In this chapter, there are a number of *classes* available for screening passengers. Each class is defined by the costs associated with purchasing, installing, and using the security devices it employs. Each class is implicitly defined by a preassigned subset of devices and a procedure through which passengers are processed prior to boarding an aircraft. Each *device* is an aviation security technology or procedure used to identify a threat. Each device screens passengers in one of three ways: by screening checked baggage, carry-on baggage, or passengers. At present, all checked baggage is screened for explosives either by an explosive detection system (EDS) or an explosive trace device (ETD). All passengers are screened with a magnometer and their carry-on baggage is screened with an X-ray machine. Selectees and their carry-on baggage are differentiated from nonselectees by undergoing hand searches by airport screening personnel. In some airports, selectees are screened with hand wands or explosive trace portals and their carry-on baggage is screened by trace devices.

Secondary screening is needed to resolve alarms in each class. In practice, the same secondary screening procedures are used to resolve alarms in all of the classes, which implies that a degree of overlap exists between the classes. However, this research focuses on the primary screening procedures associated with each class, and it assumes that there are enough resources for resolving alarms.

The chapter is organized as follows. Section 2.1 introduces the Multilevel Allocation Problem (MAP), a discrete optimization model that considers budget allocation based on class costs and shows that this problem is NP-hard. Section 2.2 reports two dynamic programming algorithms that

solve MAP in pseudo-polynomial time. Section 2.3 introduces a heuristic that provides approximate solutions to MAP. Section 2.4 reports computational results for an example using data values that reflect peak hour passenger volumes at small, medium and large airports.

2.1 Discrete Optimization Model

This section introduces MAP, a general framework for multilevel security screening, and formulates it as a discrete optimization problem. The solution to MAP provides a security plan for a peak hour of operation, and hence, also provides guidelines for any other hour of operation that has fewer passengers that require security screening. MAP is first stated as a discrete optimization problem and then formulated as an integer programming model. The objective is to assign N passengers to M classes such that the total security is maximized subject to budget and assignment constraints. The classes are defined in terms of their fixed and marginal costs, which are determined by the set of devices that define the classes. Although MAP assigns passengers to classes, it ultimately determines how the budget should be used to purchase security devices and which classes should (and should not) be used. MAP is formally stated.

The Multilevel Allocation Problem (MAP)

Given:

A set of N passengers, each of which is characterized by an assessed threat value $AT_1, AT_2, \ldots, AT_N$ with $0 < AT_i \leq 1,\ i = 1, 2, \ldots, N$,

a set of M classes,

a fixed cost associated with each class $FC_1, FC_2, \ldots, FC_M$,

a marginal cost associated with each class $MC_1, MC_2, \ldots, MC_M$,

the total budget B,

the *security level* of each class, L_i, where $0 \leq L_i \leq 1,\ i = 1, 2, \ldots, M$.

Objective: Denote passenger assignments for the N passengers to the M classes by $A_1, A_2, \ldots, A_M$, where $A_i \subseteq \{1, 2, \ldots, N\}$ represents the subset of passengers who are assigned to class i, and define

the risk level R_i of class $i = 1, 2, \ldots, M$ as the proportion of assessed threat values of the passengers assigned to class i. Hence,

$$R_i = \left(\frac{1}{\sum_{j=1}^{N} AT_j} \right) \sum_{j \in A_i} AT_j, \ i = 1, 2, \ldots, M \tag{2.1}$$

Find passenger assignments $A_1, A_2, \ldots, A_M$ such that $\bigcup_{i=1}^{M} A_i = \{1, 2, \ldots, N\}$ and $A_{i_1} \cap A_{i_2} = \emptyset$ for $i_1, \ i_2 = 1, 2, \ldots, M$, $i_1 \neq i_2$, and such that the budget constraint is satisfied (i.e., $\sum_{i=1}^{M} |A_i| \, MC_i + \sum_{\{i:|A_i|>0\}} FC_i \leq B$) and the total security is maximized (i.e., $\sum_{i=1}^{M} L_i R_i$).

The assessed threat values, the security levels and the risk levels can be set with information and data available from a prescreening system such as Secure Flight and the TSA. The assessed threat values provide risk assessment measures for each passenger (scaled between zero and one), with the assessed threat values taking on either continuous or discrete values. It is also assumed that passengers (and their assessed threat values) are independent. The assessed threat values are determined by an automated system such as Secure Flight in real-time. Note that the automated system is a computer program that determines deterministic assessed threat values based on information that passengers provide at the point of ticket purchase. It is assumed that the assessed threat values are accurate representations of the passengers' true level of risk (i.e., passengers with higher assessed threat values are more likely to be a threat than passengers with lower assessed threat values).

The security level of each class (scaled between zero and one) is based on security procedures of each device used to screen passengers in that class. In this case, the security level for class i is interpreted as the true alarm rate, the probability that a passenger who is a threat is detected given that they are assigned to class i. Likewise, the risk level for class i is defined as the conditional probability that class i contains a passenger who is a threat given that the passenger population contains a passenger who is a threat. The assessed threat value must be defined in a way such that the risk levels can be interpreted as this conditional probability. Note that the assessed threat values are determined by a system such as Secure Flight. Although the details of Secure Flight are classified, assumptions of how Secure Flight works are based on information available in the public domain, with Secure Flight used as a black box based on these assumptions. If the risk

level function of class $i = 1, 2, \ldots, M$ is interpreted as the conditional probability that there is a class i threat given that there is a threat, then the total security is the overall true alarm rate, the conditional probability that a threat is detected given that there is a passenger who is a threat. To see this, define the following events:

D = a threat is detected in the passenger population,

T = the passenger population contains a threat,

C_i = class i contains a passenger who is a threat, $i = 1, 2, \ldots, M$.

By conditioning on which class contains a threat, the total security can be expressed as

$$P(D|T) = \sum_{i=1}^{M} P(D|C_i, T)\, P(C_i|T) = \sum_{i=1}^{M} L_i\, R_i.$$

Since each of the passengers in class i are screened individually, L_i is the conditional probability of detecting a class i passenger who is a threat given that there is a threat in the population. This conditional probability is a function of the detection probabilities associated with the devices and the procedures used to screen passengers in class i. The conditional probabilities as well as the fixed and marginal costs can be obtained by the TSA when screening devices are tested and evaluated before they are incorporated into security screening systems. The objective function is linear since it captures the ratio of the risk in the classes to the total risk based on how the risk level functions are defined. The objective function is additive since all passengers are assumed to be independent, and therefore, how a passenger is screened (i.e., by which class) is independent of how other passengers are screened.

MAP is formulated as an integer program (2.2) with binary decision variables $x_{ij} = 1(0)$ if passenger j is (not) assigned to class i for $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, N$, and $y_i = 1(0)$ if there is (not) at least one passenger assigned to class $i = 1, 2, \ldots, M$.

$$\max \sum_{i=1}^{M} L_i R_i = \left(\frac{1}{\sum_{j=1}^{N} AT_j} \right) \sum_{i=1}^{M} \sum_{j=1}^{N} L_i AT_j x_{ij} \tag{2.2}$$

$$\text{subject to } \sum_{i=1}^{M} \sum_{j=1}^{N} MC_i x_{ij} + \sum_{i=1}^{M} FC_i y_i \leq B \tag{2.3}$$

$$\sum_{i=1}^{M} x_{ij} = 1,\ j = 1, 2, \ldots, N \tag{2.4}$$

$$\frac{1}{N} \sum_{j=1}^{N} x_{ij} - y_i \leq 0,\ i = 1, 2, \ldots, M \tag{2.5}$$

$$y_i \in \{0, 1\},\ i = 1, 2, \ldots, M \tag{2.6}$$

$$x_{ij} \in \{0, 1\},\ i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, N. \tag{2.7}$$

The objective is to maximize the total security (2.2), which is represented by the sum of the products of the security and risk levels. The first constraint (2.3) is the budget feasibility constraint. The second set of N constraints (2.4) ensures that each passenger is assigned to exactly one class. The third set of M constraints (2.5) ensures that the fixed costs are included for all nonempty classes. The last two sets of $M(N+1)$ constraints, (2.6) and (2.7), restrict the y_i and x_{ij} to being 0-1 binary variables.

MAP is NP-hard by a polynomial Turing reduction from the Exact k-item Knapsack Problem, which is NP-hard [13]. Note that MAP can be restricted in several ways and still remain NP-hard. First, MAP remains NP-hard when $FC_i = 0,\ i = 1, 2, \ldots, M$. Secondly, MAP remains NP-hard when the assessed threat values for all passengers are identical (i.e., when passengers are indistinguishable). Since MAP is NP-hard for a particular form of the risk levels, MAP remains NP-hard for general risk level forms $R_1, R_2, \ldots, R_M$. However, when $M = 2$, MAP is solvable in polynomial time (see Lemma 2.2).

Theorem 2.1 *MAP is NP-hard.*

Proof: The Exact k-item Knapsack Problem is a particular case of MAP when the assessed threat values for all passengers are identical and when the fixed costs are zero. □

Lemma 2.1 indicates that for any optimal solution to MAP, the passengers with higher assessed threat values are assigned to classes with higher security levels. Therefore, if n_M passengers are assigned to class M, the class with the highest security level, then the n_M passengers with the largest assessed threat values are assigned to class M. Lemma 2.1 is used to construct dynamic programming algorithms for MAP. Lemma 2.2 shows that MAP is polynomially solvable for a fixed number of classes.

Lemma 2.1 *Given an instance of MAP with* $AT_1 \leq AT_2 \leq \ldots \leq AT_N$ *and* $L_1 \leq L_2 \leq \ldots \leq L_M$*, the optimal way to assign the N passengers to the M classes given that* n_i *passengers are assigned to class* $i = 1, 2, \ldots, M$ *is to assign the first* n_1 *passengers to class 1, the next* n_2 *passengers to class 2, repeating in this manner until the last* n_M *passengers are assigned to class M.*

Proof: Let l_k, $k = 1, 2, \ldots, N$ be the security level of the class passenger k is assigned to and $n_i = |\{k|l_k = L_i\}|$ for $i = 1, 2, \ldots, M$ be the number of passengers assign to class i. Since N is in general much larger than M, each security level value may be duplicated many times. For example, if there are fifty passengers and two classes with $L_1 = 0.5$ and $L_2 = 0.9$, and the first forty passengers are assigned to class 1, then $l_1 = l_2 = ... = l_{40} = 0.5$ and $l_{41} = l_{42} = ... = l_{50} = 0.9$. The MAP objective with passenger assignments $A_1, A_2, \ldots, A_M$ is

$$\frac{1}{\sum_{j=1}^{N} AT_j} \sum_{i=1}^{M} \sum_{j \in A_i} L_i AT_j = \frac{1}{\sum_{j=1}^{N} AT_j} \sum_{j=1}^{N} l_j AT_j.$$

The proof follows from Hardy's Lemma [21], which states that if $x_1 \leq x_2 \leq \ldots \leq x_n$ and $y_1 \leq y_2 \leq \ldots \leq y_n$ are sequences of numbers, then

$$\max_{(i_1, i_2, \ldots, i_n) \in P} \sum_{j=1}^{n} x_{i_j} y_j = \sum_{j=1}^{n} x_j y_j,$$

where P is the set of all permutations of the integers $(1, 2, \ldots, n)$. □

Lemma 2.2 *MAP is polynomial solvable when there are a fixed number of classes (i.e., M is fixed).*

Proof: An enumeration algorithm considers all possible passenger assignments and returns the feasible solution with the best objective function value. Since each of the N passengers can be assigned to each of the M classes, then this enumeration algorithm considers M^N total passenger assignments.

However, consider a modified enumeration algorithm for solving MAP using Lemma 2.1 to limit the number of passenger assignments considered. The enumeration algorithm needs only to select the $M - 1$ breakpoints such that the objective function value is maximized. Therefore, the total

number of passenger assignments to consider is

$$\begin{pmatrix} N+M-1 \\ N \end{pmatrix} = \frac{(N+M-1)(N+M-2)\cdots(N+1)}{(M-1)!} = O(N^{M-1})$$

[10]. Since O(N) time is required to assign individual passengers to classes and to determine the objective function value, then this modified enumeration algorithm requires O(N^M) time, hence, is polynomial in terms of the number of passengers N for a fixed number of classes M. □

MAP does not explicitly incorporate stochastic screening times, passengers arriving in real-time, or space requirements of screening devices. Although screening time is an important operational factor, the time necessary for going through security screening procedures under normal operating conditions, including waiting time, is less than the time airlines recommend for passengers to arrive before their flights depart (generally at least sixty minutes before departure time for domestic flights). Therefore, screening times are implicitly modeled by MAP. Note that MAP does not consider passengers arriving in real-time. This is reasonable, since the TSA has access to information provided by passengers at the time of ticket purchase to determine the passengers' assessed threat values. Since most passengers purchase their tickets in advance, it is reasonable to assume that passengers arrive in a given time period prior to their flight departure time. In addition, screening devices may require significant amounts of space in airport lobbies or terminals. It is assumed that the classes are defined such that the necessary number of screening devices used by the classes is confined to the physical space available, and therefore, space requirements are implicitly captured by MAP.

2.2 Dynamic Programming Algorithms

Two dynamic programming algorithms are presented that obtain optimal solutions to MAP in O(N^2MB) and O(NMB) time, and O(NMB) space. For practical purposes, however, these algorithms require too much space to solve instances of MAP that have a large number of passengers. For both algorithms, the passengers are sorted such that $AT_1 \leq AT_2 \leq \ldots \leq AT_N$ and the security levels are sorted such that $L_1 \leq L_2 \leq \ldots \leq L_M$. This ensures that the dynamic programming algorithms assign the passengers with the highest assessed threat values to the (nonempty) class

with the largest security level. Without loss of generality, assume that $AT_1 \leq AT_2 \leq \ldots \leq AT_N$ and $L_1 \leq L_2 \leq \ldots \leq L_M$.

The dynamic programming algorithms assume that the marginal costs, the fixed costs, and the budget assume values that are multiples of a base unit α. For example, the costs may be defined in multiples of \$10, which results in $\alpha = 10$. Moreover, the intermediate budgets $\hat{B}$ may only take on a set of β discrete values such that $\hat{B} = \alpha, 2\alpha, \ldots, \beta\alpha = B$, where the recursion cannot consider other intermediate budget values. Note that if $\alpha = 1$, then $\beta = B$ in the worst case, hence $\hat{B} = 1, 2, \ldots, \beta = B$.

To describe the first dynamic programming algorithm for MAP, let $g_{m,n}(\hat{B})$ denote the optimal solution to the problem defined over the first $m = 1, 2, \ldots, M$ classes, and $n = 1, 2, \ldots, N$ passengers, with budget $\hat{B} = \alpha_1, \alpha_2, \ldots, \alpha_\beta = B$,

$$g_{m,n}(\hat{B}) = \max \left\{ \frac{1}{\sum_{j=1}^{N} AT_j} \sum_{i=1}^{m} \sum_{j=1}^{n} L_i AT_j x_{ij} \;\middle|\; \sum_{i=1}^{m} \sum_{j=1}^{n} MC_i x_{ij} + \sum_{i=1}^{m} FC_i y_i \leq \hat{B}; \right.$$

$$\left. \sum_{i=1}^{m} x_{ij} = 1,\ j = 1, 2, \ldots, n;\ \frac{1}{N} \sum_{j=1}^{n} x_{ij} - y_i \leq 0,\ i = 1, 2, \ldots, m \right\},$$

where $x_{ij} \in \{0, 1\}$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ and $y_i \in \{0, 1\}$, $i = 1, 2, \ldots, m$. The optimal solution of MAP is given by $g_{M,N}(B)$. Initially, $g_{1,n}(\hat{B}) = L_1 \sum_{j=1}^{n} AT_j$ if $FC_1 + MC_1 n \leq \hat{B}$ and $-\infty$ otherwise, $\hat{B} = \alpha_1, \alpha_2, \ldots, \alpha_\beta$, $n = 1, 2, \ldots, N$. Subsequent values of $g_{m,n}(\hat{B})$ are found by the recursion

$$g_{m,n}(\hat{B}) = \max \begin{cases} g_{m-1,n}(\hat{B}), \\ \max_{t=1,2,\ldots,T} \{ g_{m-1,n-t}(\hat{B} - FC_m - MC_m t) + L_m \sum_{j=n-t+1}^{n} AT_j \} \end{cases}$$

where $T = \min\{n, \lfloor \frac{\hat{B} - FC_m}{MC_m} \rfloor\}$. At each step of the recursion, either no passengers are assigned to class m, in which case the fixed cost is not assessed against the budget, or between 1 and T unassigned passengers with the highest assessed threat values are assigned to class m. Since $\lfloor \frac{\hat{B} - FC_m}{MC_m} \rfloor = N$ in the worst case, each step in the recursion can call at most N other recursions, resulting in a total time bound of $\mathrm{O}(N^2 M\, B)$ in the worst case. Storing the values of $g_{m,n}(\hat{B})$ requires space bound $\mathrm{O}(M\, N\, B)$.

To describe the second dynamic programming algorithm, let $g_{m,n}(\hat{B})$ be defined as in the first dynamic programming algorithm , and let $f_{m,n}(\hat{B})$, $m = 1, 2, \ldots, M$, $n = 1, 2, \ldots, N$, $\hat{B} = \alpha_1, \alpha_2, \ldots, \alpha_\beta = B$ be the optimal solution over the first m classes and n passengers with budget $\hat{B}$ given that at least one passenger is assigned to class m:

$$\begin{aligned} f_{m,n}(\hat{B}) \quad = \quad & \max \left\{ \frac{1}{\sum_{j=1}^{N} AT_j} \sum_{i=1}^{m} \sum_{j=1}^{n-1} L_i AT_j x_{ij} + L_m AT_n \;\middle|\; \sum_{i=1}^{m} \sum_{j=1}^{n-1} MC_i x_{ij} + \right. \\ & \sum_{i=1}^{m-1} FC_i y_i \leq \hat{B} - FC_m - MC_m; \sum_{i=1}^{m} x_{ij} = 1,\ j = 1, 2, \ldots, n-1; \\ & \left. \frac{1}{N} \sum_{j=1}^{n-1} x_{ij} - y_i \leq 0, i = 1, 2, \ldots, m-1 \right\} \end{aligned}$$

with $x_{ij} \in \{0, 1\}$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n-1$ and $y_i \in \{0, 1\}$, $i = 1, 2, \ldots, m-1$. The optimal solution of MAP is again given by $g_{M,N}(B)$. Initially, $f_{1,n}(\hat{B}) = g_{1,n}(\hat{B}) = L_1 \sum_{j=1}^{n} AT_j$ if $FC_1 + MC_1 n \leq \hat{B}$ and $-\infty$ otherwise. Additionally, $f_{m,1}(\hat{B}) = -\infty$ for $m = 1, 2, \ldots, M$, $\hat{B} < FC_m + MC_m$, and $f_{m,1}(\hat{B}) = L_m AT_1$ for $m = 1, 2, \ldots, M$, $\hat{B} \geq FC_m + MC_m$. Moreover, $f_{m,n}(\hat{B}) = -\infty$ if $FC_m + MC_m < \hat{B}$ for $m = 1, 2, \ldots,$ $n = 1, 2, \ldots, N$. Subsequent values of $f_{m,n}(\hat{B})$ are found by the recursion

$$f_{m,n}(\hat{B}) = \max \begin{cases} g_{m-1,n-1}(\hat{B} - FC_m - MC_m) + L_m AT_n \\ f_{m,n-1}(\hat{B} - MC_m) + L_m AT_n \}. \end{cases} \tag{2.8}$$

Subsequent values of $g_{m,n}(\hat{B})$ are found by the recursion

$$g_{m,n}(\hat{B}) = \max\{g_{m-1,n}(\hat{B}),\ f_{m,n}(\hat{B})\}. \tag{2.9}$$

At each step of (2.9), either no passengers are assigned to class m, in which case the fixed cost of class m is not assessed against the budget, or at least one passenger is assigned to class m by (2.8). This dynamic programming algorithm runs in $O(N\,M\,B)$ time. The values $g_{m,n}(\hat{B})$ and $f_{m,n}(\hat{B})$ require $O(N\,M\,B)$ space. When the dynamic programming algorithm is complete, the number of passengers assigned to each class can be determined from iterating back from $g_{M,N}(B)$ and $f_{M,N}(B)$. Once the number of passengers assigned to each class is known, the passenger partitions

are determined by Lemma 2.1.

2.3 Greedy Heuristic

This section introduces the Two Class Greedy Heuristic (2GH) that obtains approximate solutions to MAP in polynomial time. 2GH first finds the number of passengers assigned to each class and then constructs the exact passenger assignments. 2GH obtains the solution to MAP with the highest objective function value using no more than two classes. To describe 2GH, let $C^{N^+} = \left\{i : \left\lfloor \frac{B-FC_i}{MC_i} \right\rfloor \geq N\right\}$, the subset of classes such that the budget is sufficiently large for each class to screen all N passengers, and let $C^{N^-} = \left\{i : \left\lfloor \frac{B-FC_i}{MC_i} \right\rfloor < N\right\}$, the remaining subset of classes. Note that C^{N^-} and C^{N^+} are mutually exclusive and exhaustive subsets of $\{1, 2, \ldots, M\}$. Two trivial special cases are $C^{N^+} = \emptyset$, when there is no feasible solution (i.e., the budget is not large enough to assign all N passengers to the M classes), and $C^{N^-} = \emptyset$, when the optimal solution value is equal to the largest security level (i.e., the budget is sufficient to assign all N passengers to the class with the largest security level).

To find the solution to MAP with the highest objective function value using no more than two classes, 2GH selects the solution from a set of candidate solutions with the highest objective function value. The candidate solutions are selected from the set of feasible solutions to MAP, where the remaining (non-candidate) feasible solutions can be pruned by assigning individual passengers to classes (see Lemma 2.1). The set of candidate solutions are obtained as follows: 2GH begins by partitioning the classes into C^{N^-} and C^{N^+} and finding the best solution using a single class. The optimal solution using exactly one class assigns all N passengers to the class in C^{N^+} with the largest security level. 2GH then obtains candidate solutions by considering solutions using exactly two classes, with one class in C^{N^-} and one in C^{N^+}, where each such pair of classes results in at most one candidate solution. The best solution using classes $i^+ \in C^{N^+}$ and $i^- \in C^{N^-}$ assigns the maximum number of passengers into the class with the highest security level such that the assignment and budget constraints are satisfied (i.e., the solution is feasible). Once the number of passengers assigned to each class is known, individual passengers are assigned to the two classes (see Lemma 2.1). Only classes $i^- \in C^{N^-}$ and $i^+ \in C^{N^+}$ such that $L_{i^+} < L_{i^-}$ need to be considered; otherwise the resulting solutions would be no better than the best solution using a single class.

In the 2GH pseudocode, z_h is the current best objective function value, the vector x of length MN is its associated passenger assignment, and B_h is the interim cost of this passenger assignment. For each candidate solution considered, z'_h is the objective function value, $\lfloor n_{i^-} \rfloor$ and $\lceil n_{i^+} \rceil$ are the number of passengers assigned to $i^- \in C^{N^-}$ and $i^+ \in C^{N^+}$, respectively, and the vector x' is the passenger assignment.

2GH executes in $O(\max\{N \log N, M^2 N\})$ time. The initial sorting of the assessed threat values and the partitioning of the classes into C^{N^+} and C^{N^-} requires $O(M + N \log N)$ time. Note that 2GH considers all combinations of two classes such that one of these classes is in C^{N^+} and the other class is in C^{N^-}. Therefore, 2GH obtains $|C^{N^-}|\,|C^{N^+}| \leq M^2$ candidate solutions in the worst case, and requires $O(N)$ time to compute the objective function value for each candidate solution. If the assessed threat values are identical, then 2GH obtains a solution in $O(M^2)$ time since the assessed threat values do not need to be sorted and the objective function values are computed in constant time.

Algorithm 1 The Two Class Greedy Heuristic

Sort assessed threat values such that $AT_1 \leq AT_2 \leq \ldots \leq AT_N$
Partition the set of classes into C^{N^-} and C^{N^+}
$z_h = \max\{L_i : i \in C^{N^+}\}$
$x_{ij} = 0,\ i = 2, \ldots, M,\ j = 1, 2, \ldots, N;\ x_{C_1^+ j} = 1, j = 1, 2, \ldots, N$
$B_h = N \cdot MC_{C_1^+} + FC_{C_1^+}$
for all $i^+ = C^{N^+}$, $i^- = C^{N^-}$ **do**
  Solve the linear equations:
  (1) $(MC_{i^+})n^+ + (MC_{i^-})n^- = B - FC_{i^+} - FC_{i^-}$
  (2) $n^+ + n^- = N$
  if $0 < n^- < N$ **then**
    compute z'_h, x' based on putting the $\lceil n^+ \rceil$ passengers with the lowest AT values in class i^+ and the remaining $\lfloor n^- \rfloor$ passengers with the highest AT values in class i^-
    if $z'_h > z_h$ **then**
      $z_h = z'_h,\ x \leftarrow x'$
      $B_h = FC_{i^+} + FC_{i^-} + \lceil n^+ \rceil MC_{i^+} + \lfloor n^- \rfloor MC_{i^-}$
    end if
  end if
end for
return $x,\ z_h,\ B_h$

Theorem 2.2 shows that 2GH is (1/2)-optimal for the particular case of MAP when passengers are indistinguishable.

Theorem 2.2 *2GH always obtains solutions to MAP that are at least 1/2 of the optimal solution value when passengers have identical assessed threat values.*

Proof: When all of the passengers have identical assessed threat values, then the particular instance of MAP can be formulated as an instance of the k-item Integer Knapsack Problem with Set-up Weights (kIKPSW) (see Chapter 6). The kIKPSW Greedy Heuristic, $H_k^{1/2}$, finds the best solution using no more than two item types, where such solutions are always within at least 1/2 of the optimal solution value (see Theorem 6.6). 2GH is the same as $H_k^{1/2}$ when the passengers have identical assessed threat values. □

2.4 Computational Results

This section provides computational results for MAP. The results incorporate data extracted from the Official Airline Guide (OAG) for the United States domestic flights of a single airline carrier. The data provided by the OAG includes the set of flights, the number of available seats on each flight, and the departure time of each flight. It is assumed that all passengers have exactly one checked and one carry-on bag.

A total of 270 scenarios are considered, where these scenarios were designed using three passenger sets (i.e., the number of passengers), three assessed threat distributions, three types of classes, and ten budget values. The three passenger sets are defined based on OAG data, which consider 10, 30, and 60 minute windows of time. To find the number of passengers in these time segments, passengers are assumed to arrive randomly according to a uniform distribution between 30 and 90 minutes prior to the departure time of each flight. This arrival interval is recommended by airlines for domestic flights (e.g., see www.nwa.com, www.ual.com, www.aa.com, and www.delta.com). Moreover, each flight is assumed to have an enplanement rate of 80% (i.e., the number of passengers divided by the number of available seats). The data set is chosen based on finding the largest expected number of arriving passengers in 10, 30, and 60 minute windows, resulting in 1230, 3690, and 6200 passengers, respectively. The flights associated with each group of passengers can be identified and grouped together resulting in three subsets of flights for the different scenarios.

The scenarios consider three hypothetical assessed threat value distributions based on information from the TSA. Initially, all passengers are assumed to be identical resulting in a degenerate distribution with all assessed values equal to one. This corresponds to the case when no information is known about the passengers (call this Type I). The next two assessed threat distributions are

taken to be exponential distributions with means 1/8 and 1/16, respectively, that are truncated to allow only values less than one [72]; call these Type II and Type III, respectively. Note that the Type II and Type III assessed threat distributions result in approximately 80% of the passengers having assessed threat values less than 0.2 and 0.1, respectively. These assessed threat distributions model situations where few passengers are perceived to be potential risks, hence most passengers have low assessed threat values.

Table 2.1 contains the security screening device data used for all the scenarios. It is divided into three areas: checked baggage, passenger, and carry-on baggage screening devices. The device values are estimated using information available in the public domain [12, 89]. The yearly costs are computed based on the purchase costs, the expected lifetime of the device, and the yearly maintenance costs. In Table 2.1, the unadjusted fixed cost FC' is the fixed cost per hour per 1000 passengers, and it is based on the yearly fixed costs divided by the hours of operation per year (360 days a year, 6 peak hours per day), normalized by the capacity.

Table 2.1: Security Device Data

Device Category	Device Type	False Clear	FC′	MC	Units/hour
Checked Baggage	EDS	0.12	4.167	1.00	125
	Open Bag Trace (OT)	0.15	1.199	0.83	28
Passenger	Metal Detector (MD)	0.30	0.051	0.28	90
	Hand Wand Inspection (HW)	0.20	0.009	1.25	20
Carry-on Baggage	X-ray Machine (XR)	0.20	0.720	0.28	90
	Detailed Hand Search (DHS)	0.20	0	1.25	20
	Open Bag Trace/ Detailed Search (OTDS)	0.15	1.199	1.29	18

EDSs screen checked baggage for the presence of materials commonly used in bombs. An open bag trace search tests for the presence of chemicals commonly used in bombs by sampling the inside of a bag with a swab. Metal detectors (i.e., magnometers) and hand wand inspections screen passengers for concealed metal objects such as guns and knives. An X-ray machine screens carry-on baggage for these items. A detailed hand search involves airport screening personnel searching for concealed weapons and bombs manually. An open bag trace test can be combined with a detailed hand search.

Three cases are considered, consisting of three, five, and eight classes. The three-class scenarios are motivated by the TSA CAPPS II description, while allowing all passengers to board an aircraft [80]. The five-class and eight-class scenarios correspond to environments with passengers of varying

perceived risk levels using existing security screening devices. The marginal costs, fixed costs, and security level associated with each class for all three class scenarios are summarized in Table 2.2. The costs given are in (US) dollars. These are computed from the values associated with each of the devices in each class and the number of passengers.

The marginal costs are computed by summing the marginal costs of the devices used by the class. The fixed costs are computed by summing the fixed costs of the devices used by the class and scaling to account for the length of the time window and the number of passengers. The fixed cost associated with open trace is only assessed once, even if both types of open trace are used in the class. Furthermore, if there are M classes, then the fixed cost values are divided by M. This is done to compensate for the fact that not all passengers are screened by all devices and that some classes share security screening devices.

The security levels of the classes for the example are the overall true alarm rate, the probability that a threat is detected given that there is a threat. A passenger who is a threat is assumed to be detected if at least one security device gives an alarm response. Additionally, it is assumed that it is equally likely for a threat to be in a checked bag, carry-on bag, or on a person. This assumption is reasonable since no data exists that suggests a distribution among these means of attack.

Table 2.2: Security Class Costs and Security Level Values

Class	Devices	FC N=1230	FC N=3690	FC N=6200	MC	L
1	EDS, MD, XR	67.49	202.47	340.19	1.56	0.793
2	EDS, MD, HW, XR DHS	67.62	202.85	340.83	2.81	0.927
3	EDS, OT, MD, HW, XR, OTDS	93.10	279.30	469.29	4.93	0.964
1	MD, XR	18.98	56.94	95.68	0.56	0.500
2	EDS, MD, XR	121.48	364.44	612.35	1.56	0.793
3	EDS, MD, XR, DHS	121.48	364.44	612.35	2.81	0.847
4	OT, MD, HW, XR, DHS	48.70	146.10	245.47	2.64	0.917
5	EDS, OT, MD, HW, XR, OTDS	167.58	502.74	844.72	4.93	0.964
1	MD, XR	23.73	71.18	119.60	0.56	0.500
2	MD, HW, XR	24.01	72.03	121.03	1.81	0.580
3	EDS, MD, XR	151.85	455.56	765.43	1.56	0.793
4	EDS, MD, HW, XR	151.85	455.56	765.43	2.81	0.847
5	EDS, MD, XR, DHS	152.14	456.41	766.87	2.81	0.873
6	OT, MD, HW, XR, DHS	60.87	182.62	306.84	3.89	0.917
7	OT, MD, HW, XR, OTDS	81.35	244.05	410.06	3.93	0.920
8	EDS, OT, MD, HW, XR, OTDS	209.48	628.43	1055.90	4.93	0.964

Each scenario is addressed using the 2GH implemented in Matlab. All the scenarios are also for-

Table 2.3: Solutions for Three-Class Scenarios

		Type I			*Type II*			*Type III*		
		2GH	IP	IP CPU	2GH	IP	IP CPU	2GH	IP	IP CPU
N	B ($)	value	value	time (sec)	value	value	time (sec)	value	value	time (sec)
1230	2202.47	**0.800**	0.800	1	**0.820**	0.820	2	**0.819**	0.8192	2
1230	2646.91	**0.839**	0.839	1	**0.889**	0.889	3	**0.887**	0.8868	3
1230	3091.36	**0.877**	0.877	1	0.917	0.918	19	0.916	0.916	32
1230	3535.80	**0.916**	0.916	1	0.927	0.935	3	0.927	0.934	5
1230	3980.25	**0.931**	0.931	1	0.940	0.946	3	0.940	0.945	3
1230	4424.69	**0.938**	0.938	1	0.951	0.953	3	0.950	0.952	3
1230	4869.14	**0.944**	0.944	1	0.957	0.958	3	0.957	0.958	3
1230	5313.58	**0.950**	0.950	1	0.961	0.962	3	0.961	0.961	3
1230	5758.03	**0.956**	0.956	1	0.963	0.963	2	0.963	0.963	2
1230	6202.47	**0.963**	0.963	1	**0.964**	0.964	2	**0.964**	0.964	2
3690	6407.41	**0.794**	0.794	1	**0.800**	0.800	9	**0.800**	0.8008	9
3690	7762.97	**0.834**	0.834	1	**0.883**	0.883	10	**0.882**	0.882	13
3690	9118.52	**0.873**	0.873	1	0.915	0.916	9	0.914	0.916	9
3690	10474.08	**0.912**	0.912	1	0.926	0.934	9	0.926	0.933	12
3690	11829.63	**0.931**	0.931	1	0.940	0.945	10	0.940	0.945	9
3690	13185.19	**0.937**	0.937	1	0.951	0.953	9	0.951	0.953	10
3690	14540.74	**0.944**	0.944	1	0.957	0.958	11	0.957	0.958	10
3690	15896.30	**0.950**	0.950	1	0.961	0.962	8	0.961	0.961	10
3690	17251.85	**0.957**	0.957	1	0.963	0.963	10	0.963	0.963	9
3690	18607.41	**0.963**	0.963	1	**0.964**	0.964	8	**0.964**	0.964	8
6200	10720.57	**0.794**	0.794	1	**0.796**	0.796	34	**0.796**	0.796	7359
6200	13042.79	**0.834**	0.834	1	**0.882**	0.882	48	**0.881**	0.881	105
6200	15365.01	**0.874**	0.874	1	**0.915**	0.915	73	**0.914**	0.914	75
6200	17687.24	**0.914**	0.914	1	0.926	0.933	226	0.926	0.933	45
6200	20009.46	**0.931**	0.931	1	0.940	0.945	67	0.940	0.945	36
6200	22331.68	**0.938**	0.938	1	0.951	0.953	61	0.951	0.953	31
6200	24653.90	**0.944**	0.944	1	0.957	0.958	67	0.957	0.958	34
6200	26976.13	**0.951**	0.951	1	0.961	0.962	54	0.961	0.962	32
6200	29298.35	**0.957**	0.957	1	0.963	0.963	61	0.963	0.963	24
6200	31620.57	**0.964**	0.964	1	**0.964**	0.964	31	**0.964**	0.964	13

mulated as integer programming models (IPs) and solved using CPLEX 7.0. All the computational experiments were executed on a Pentium III 550 MHz processor with 1048 MB of RAM. Tables 2.3, 2.4, and 2.5 contain the 2GH and IP values for the scenarios, as well as the CPU times for the IPs. For each IP, CPLEX was halted after 170,000 CPU seconds (approximately 2 CPU days) if it had not found an optimal solution. This value is sufficiently long to give CPLEX a reasonable amount of time to solve the problems. If the IP was not solved to optimality, an asterisk (*) is listed as its IP value. Therefore, all the IP values listed are optimal values. All 90 three-class scenarios finished in the allotted time, whereas 87 and 72 five-class and eight-class scenarios, respectively, finished in the allotted time. The Type II and Type III solutions always have larger objective function values than the corresponding Type I solution with the same number of passengers and budget value. This suggests that models that incorporate such information about passengers are more effective than models which assume that passengers are indistinguishable. For small budget values, the Type II and Type III solutions have significantly higher objective values than the corresponding Type I solution. As the budget increases, these differences become less pronounced, which suggests that when a large budget is available, it is less critical to distinguish between passengers (since nearly all passengers can then be assigned to the class with the highest (and most costly) security level).

For all of the Type I IPs, the scenarios with indistinguishable passengers finished in under two CPU seconds regardless of the number of classes or the number of passengers. The Type II and Type III IPs took no less than two CPU seconds to complete. Of the 90 three-class IPs, 89 finished in less than one CPU hour, and 81 finished in less than one CPU minute. Of the 87 five-class IPs that finished, 74 finished in less than one CPU hour, and 47 finished in less than one CPU minute. Of the 72 eight-class IPs that finished, 60 finished in less than one CPU hour and 39 finished in less than one CPU minute. In general, the IPs with relatively small budget values took longer to solve, while the IPs that had larger budget values took the least amount of computing time, particularly when the budgets were large enough to assign nearly all passengers to the class with the highest security level.

In all 90 Type I scenarios and in 34 of 78 (finished) Type II and 33 of 81 (finished) Type III scenarios, the 2GH solution is identical to the optimal solution. The scenarios where the 2GH value and the IP value matched are in boldface in Tables 2.3, 2.4, and 2.5. Typically, scenarios with either small or large budget values had the same 2GH and IP values, where the optimal solutions used two classes. When the 2GH and IP values were different, the IP solutions never used more than

Table 2.4: Solutions for Five-Class Scenarios

		Type I			*Type II*			*Type III*		
		2GH	IP	IP CPU	2GH	IP	IP CPU	2GH	IP	IP CPU
N	B ($)	value	value	time (sec)	value	value	time (sec)	value	value	time (sec)
1230	800	**0.507**	0.507	1	**0.533**	0.533	10648	**0.533**	0.533	16884
1230	1400	**0.636**	0.636	1	**0.753**	0.753	86201	**0.746**	0.746	79460
1230	2000	**0.779**	0.779	1	**0.852**	0.852	3214	**0.847**	0.847	19554
1230	2600	**0.841**	0.841	1	**0.900**	0.900	139	0.898	0.898	44
1230	3200	**0.898**	0.898	1	0.9166	0.922	190	0.916	0.921	17
1230	3800	**0.923**	0.923	1	0.9344	0.941	20	0.934	0.940	8
1230	4400	**0.933**	0.933	1	0.9503	0.952	18	0.949	0.951	27
1230	5000	**0.943**	0.943	1	0.9585	0.959	1555	0.958	0.958	7
1230	5600	**0.953**	0.953	1	0.9626	0.963	286	0.962	0.962	6
1230	6200	**0.963**	0.963	1	**0.964**	0.964	6	**0.964**	0.964	3
3690	2400	**0.507**	0.507	1	**0.535**	0.535	6618	**0.535**	0.535	3597
3690	4200	**0.636**	0.636	1	**0.752**	0.752	159	**0.748**	0.748	469
3690	6000	**0.779**	0.779	1	**0.850**	0.850	240	**0.849**	0.849	808
3690	7800	**0.841**	0.841	1	**0.890**	0.900	157692	**0.899**	0.899	84619
3690	9600	**0.898**	0.898	1	0.917	0.922	158	0.916	0.921	519
3690	11400	**0.923**	0.923	1	0.934	0.940	51	0.934	0.940	126
3690	13200	**0.933**	0.933	1	0.950	0.952	130	0.950	0.951	109
3690	15000	**0.943**	0.943	1	0.958	0.959	1310	0.958	0.959	1029
3690	16800	**0.953**	0.953	1	**0.963**	0.963	30	**0.962**	0.9625	36
3690	18600	**0.963**	0.963	1	**0.964**	0.964	15	**0.964**	0.964	10
6200	4000	**0.506**	0.506	1	**0.531**	0.531	6934	**0.531**	0.531	16624
6200	7033.33	**0.635**	0.635	1	0.749	*		**0.746**	0.746	1165
6200	10066.67	**0.778**	0.778	1	0.848	*		**0.848**	0.848	4233
6200	13100	**0.841**	0.841	1	**0.899**	0.899	78617	0.898	*	
6200	16133.33	**0.898**	0.898	1	0.917	0.922	153615	0.916	0.921	354
6200	19166.67	**0.923**	0.923	1	0.935	0.940	117	0.934	0.940	128
6200	22200	**0.933**	0.933	1	0.950	0.952	91	0.950	0.951	112
6200	25233.33	**0.943**	0.943	1	0.958	0.959	85	0.958	0.959	115
6200	28266.67	**0.953**	0.953	1	0.963	0.963	87	**0.962**	0.962	87
6200	31300	**0.963**	0.963	1	**0.964**	0.964	36	**0.964**	0.964	28

Table 2.5: Solutions for Eight-Class Scenarios

		Type I			*Type II*			*Type III*		
		2GH	IP	IP CPU	2GH	IP	IP CPU	2GH	IP	IP CPU
N	B ($)	value	value	time (sec)	value	value	time (sec)	value	value	time (sec)
1230	800	**0.503**	0.503	1	0.513	*		0.513	*	
1230	1400	**0.627**	0.627	1	**0.736**	0.736	114570	**0.732**	0.732	14505
1230	2000	**0.770**	0.770	1	0.793	0.814	741	0.792	0.810	2054
1230	2600	**0.813**	0.813	1	0.850	0.861	132	0.849	0.858	197
1230	3200	**0.844**	0.844	1	0.889	0.897	2075	0.886	0.893	59
1230	3800	**0.877**	0.877	1	0.921	0.923	133	0.917	0.920	47
1230	4400	**0.897**	0.897	1	0.941	0.942	4548	0.938	0.940	277
1230	5000	**0.920**	0.920	1	0.953	0.954	22	0.952	0.952	48
1230	5600	**0.937**	0.937	1	**0.961**	0.961	1075	**0.960**	0.960	274
1230	6200	**0.959**	0.959	1	**0.964**	0.964	7	**0.964**	0.964	8
3690	2500	**0.506**	0.506	1	0.530	*		0.531	*	
3690	4300	**0.636**	0.636	1	**0.741**	0.741	128020	0.740	*	
3690	6100	**0.778**	0.778	1	0.795	*		0.793	0.815	83785
3690	7900	**0.815**	0.815	1	0.853	*		0.852	0.861	4852
3690	9700	**0.846**	0.846	1	0.891	0.898	732	0.890	0.896	102
3690	11500	**0.878**	0.878	1	0.921	*		0.920	0.922	3945
3690	13300	**0.898**	0.898	1	0.941	*		0.940	0.941	1794
3690	15100	**0.920**	0.920	1	0.954	0.954	2731	0.953	0.953	2285
3690	16900	**0.938**	0.938	1	**0.961**	0.961	107475	0.961	*	
3690	18700	**0.960**	0.960	1	**0.964**	0.964	26	**0.964**	0.964	33
6200	4000.00	**0.503**	0.503	1	**0.513**	0.513	1446	0.513	*	
6200	7055.56	**0.628**	0.628	1	**0.734**	0.734	799	**0.733**	0.733	990
6200	10111.11	**0.772**	0.772	1	0.792	*		0.792	*	
6200	13166.67	**0.814**	0.814	1	0.851	*		0.850	*	
6200	16222.22	**0.845**	0.845	1	0.890	*		0.888	0.894	8634
6200	19277.78	**0.878**	0.878	1	0.920	0.922	7919	0.920	*	
6200	22333.33	**0.898**	0.898	1	0.940	0.941	3597	0.940	0.941	6890
6200	25388.89	**0.920**	0.920	1	0.953	*		0.953	0.953	1272
6200	28444.44	**0.939**	0.939	1	**0.961**	0.961	14360	**0.961**	0.961	1321
6200	31500.00	**0.961**	0.961	1	**0.964**	0.964	58	**0.964**	0.964	108

Table 2.6: Average 2GH CPU Times (seconds)

M	N	Type I	Type II	Type III
3	1230	0.031	0.033	0.034
3	3690	0.095	0.089	0.092
3	6200	0.14	0.14	0.15
5	1230	0.064	0.067	0.062
5	3690	0.19	0.18	0.18
5	6200	0.30	0.30	0.31
8	1230	0.14	0.15	0.15
8	3690	0.44	0.43	0.44
8	6200	0.72	0.73	0.73

three classes.

Table 2.6 contains the average CPU time to execute 2GH for each of the sets of scenarios. All of the average 2GH CPU times for a given number of classes, number of passengers, and assessed threat distribution are less than 0.73 CPU seconds, with the 2GH never taking longer than 1.3 CPU seconds to identify a single approximate solution. Moreover, for a given number of classes and passengers, the 2GH CPU time remained approximately the same across the Type I, II, and III assessed threat distributions. This can be contrasted to the corresponding IPs, which were solved quickly for Type I scenarios, and generally took longer for the Type II and III scenarios.

The quality of the 2GH solutions can be measured by the *relative effectiveness measure*

$$\Gamma = \frac{z^h - z_0}{z - z_0},$$

where z^h is the 2GH solution objective function value, z_0 is the objective function value of the worst possible feasible solution, and z is the objective function value of the optimal solution. If the relative effectiveness measure is one, then 2GH obtains an optimal solution, while if it is zero, then 2GH obtains the worst possible feasible solution. This measure is used, rather than z^h/z since the worst possible feasible solution may have an objective function value close to the optimal. In such cases, 2GH may yield the worst possible feasible solution (hence $\Gamma = 0$), while the absolute effectiveness measure would be close to one. For practical purposes, this would not be an issue. However, to obtain a clearer measure of the heuristic's effectiveness, the relative effectiveness measure circumvents this situation.

Figures 2.1, 2.2, and 2.3 depict the relative effectiveness measure of the 2GH solutions for the three-class, five-class, and eight-class scenarios, respectively. Several relative effective measure values could not be computed for the five-class and eight-class solutions since their corresponding

IPs did not finish in the allotted time, hence several points are missing from Figures 2.2 and 2.3. Note that the Type I scenarios are omitted since the relative effectiveness measures for these scenarios are all one. The Type II and Type III scenarios often have relative effectiveness measures of one when the budget is either small or large. The relative effectiveness measure is never less than 0.943 for the three-class scenarios, 0.986 for the five-class scenarios, and 0.930 for the eight-class scenarios, which indicates that the 2GH solutions are close to the optimal values when using just two classes.

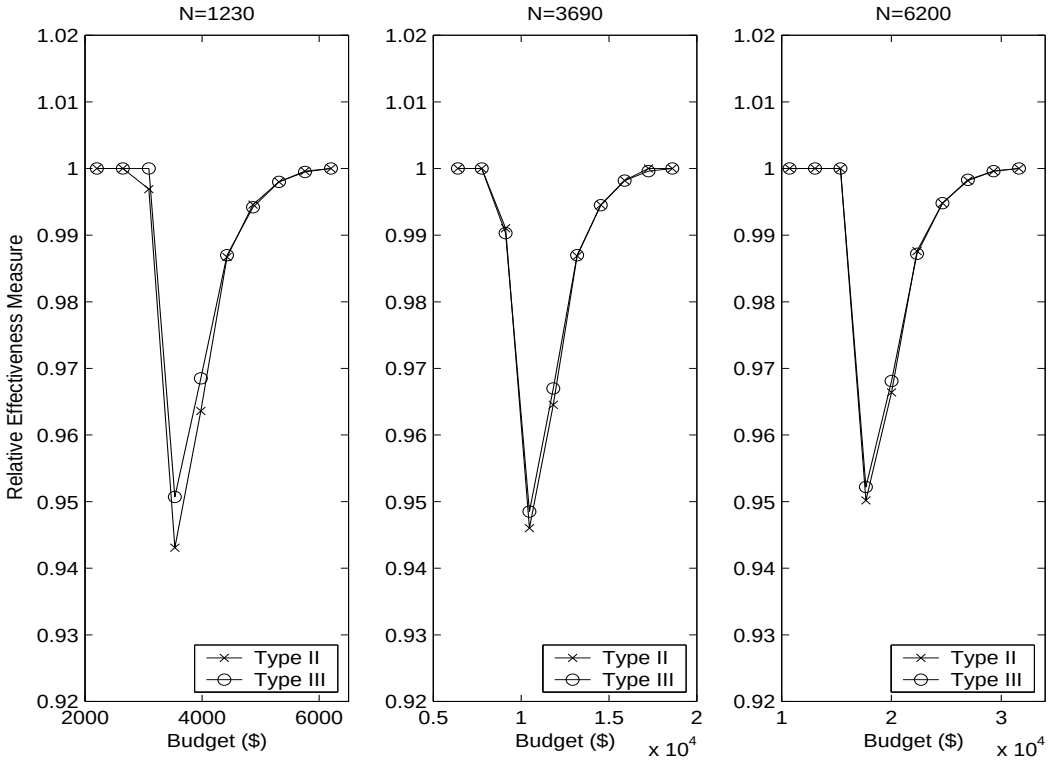

Figure 2.1: Relative Effectiveness Measure for Three-Class Examples

2.5 Conclusions

Passenger and baggage screening is a critical component of any aviation security system operation. This chapter introduces MAP, a framework to model passenger and baggage screening systems. MAP models each security class in terms of its marginal and fixed costs, and security level, where passenger assignment and budget constraints must be satisfied. MAP is formulated as an integer programming model. MAP is shown to be NP-hard. Furthermore, 2GH is introduced to obtain

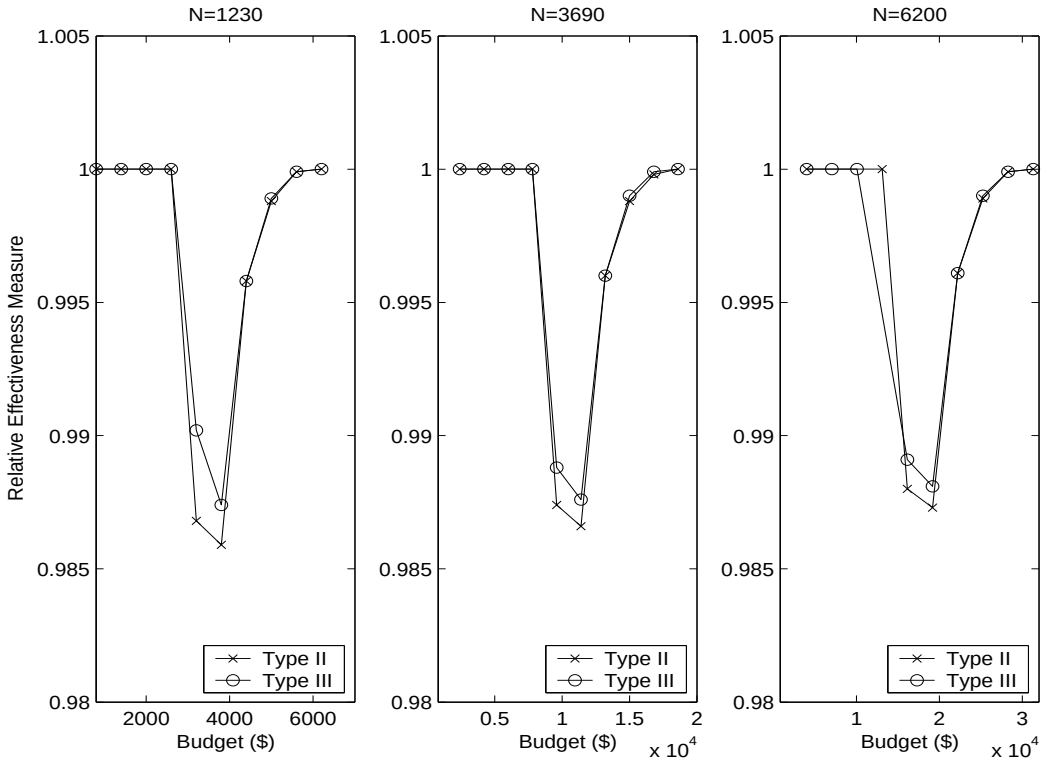

Figure 2.2: Relative Effectiveness Measure for Five-Class Examples

approximate solutions to MAP in $O(M^2N)$ time with an additional $O(N \log N)$ time for the initial sorting of passengers. In practice, 2GH obtains solutions to MAP in real-time.

Data extracted from the OAG and hypothetical data based on information provided by the TSA was used to construct a total of 270 scenarios for MAP. The optimal (IP) and 2GH objective function values were computed for each of these scenarios, where twenty-one of the IPs did not terminate with an optimal solution in the allotted computing time (three of which are five-class scenarios and eighteen of which are eight-class scenarios). In many cases, 2GH obtained the optimal solution value, including all of the Type I scenarios, 34 of the 78 (completed) Type II scenarios, and 33 of the 81 (completed) Type III scenarios. The relative effectiveness measures for the 2GH are greater than 0.940 in the three-class scenarios, 0.986 for the five-class scenarios, and greater than 0.930 for the eight-class scenarios. One limitation of the 2GH is that it obtains solutions that use no more than two classes. However, the optimal solutions for all of the Type I scenarios use two classes, and none of the completed optimal solutions for the Type II and Type III scenarios use more than three classes. This suggests that even when many classes are available, it may be more effective (from a security standpoint) to use fewer classes. This implication is desirable

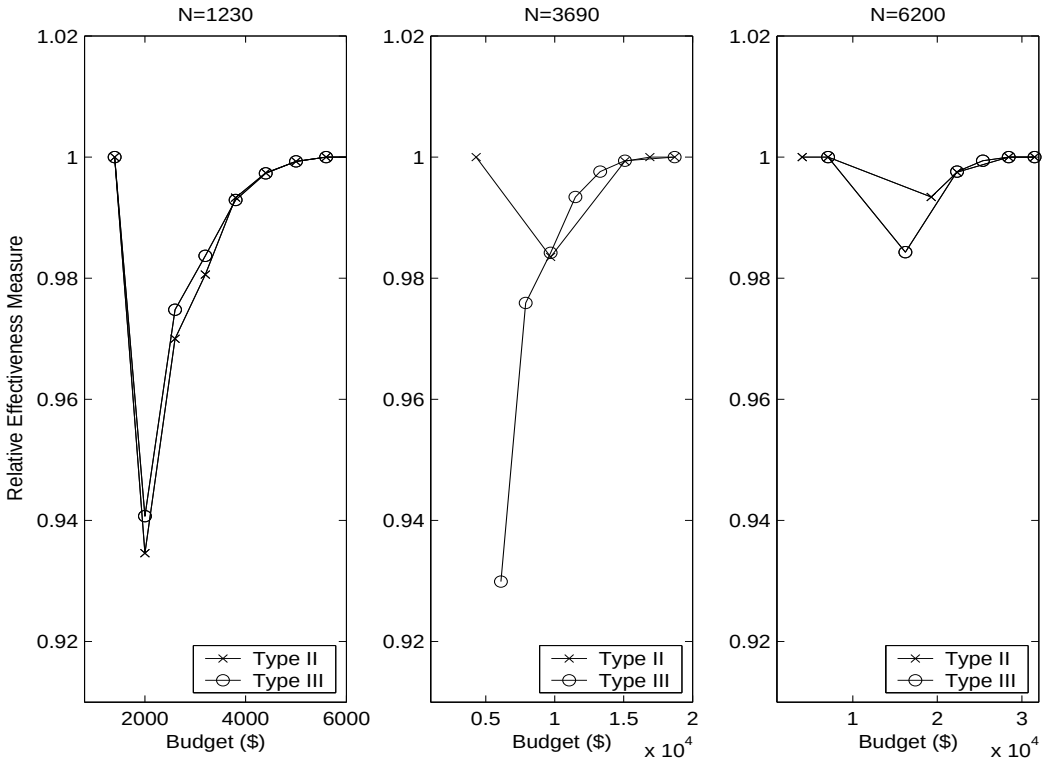

Figure 2.3: Relative Effectiveness Measure for Eight-Class Examples

on a practical level since security personnel need to be trained to be fluent with fewer security procedures. Therefore, any security system design that is simple to implement and easy to operate is of added value.

Chapter 3

The Multilevel Passenger Screening Problem

This chapter introduces the Multilevel Passenger Screening Problem (MPSP), which models multilevel passenger screening strategies using discrete optimization models and integer programs [51]. Multilevel screening problems are motivated by MAP [52]. In both MAP and MPSP, there are a number of classes available for screening passengers. In MAP, the classes are defined by the costs associated with purchasing, installing, and using the security devices it employs. This can be contrasted with MPSP, in which each class is defined by the capacities of the security devices it employs. The primary contribution of this research is to develop models for designing multilevel screening security systems, and show how these models can be used to provide insights into the operation and performance of such systems.

This chapter is organized as follows. Section 3.1 introduces MPSP and formulates it as an integer programming model based on the capacity of the devices associated with each of the classes. Section 3.2 provides several results for the integer and linear programming formulations of MPSP. Section 3.3 describes an example using passenger volume and flight data extracted from the Official Airline Guide and discusses computational results for 40 scenarios based on the example. Section 3.4 provides concluding comments and directions for future research.

3.1 Integer Programming Model

This section introduces MPSP as a framework for multilevel screening strategies. MPSP is formulated as an integer programming model, where the objective is to assign N passengers to M classes such that the total security is maximized subject to assignment and device capacity constraints, where each class is defined in terms of the devices it uses. Passengers are screened by a sequence of devices associated with the class to which they have been assigned. This sequence of devices is assumed to be fixed in advance, based on the procedures set by the TSA. Although MPSP assigns passengers to classes, it implicitly determines which devices are critical for passenger screening

operations (i.e., which devices operate at capacity) as well as which classes should (and should not) be used.

Note that each device gives one of two possible responses: alarm or clear. In addition, the system gives one of two possible outcomes: alarm or clear, and each outcome is a function of the device outcomes and can be defined in one of several ways [38, 39]. Each passenger is either a threat or a nonthreat. Ideally, the system yields a clear response for all of the nonthreat passengers and yields an alarm response for all of the threat passengers. Although it is not known in advance whether a given passenger will trigger an alarm response, it is assumed that there are procedures in place for resolving alarms and that enough resources available to resolve all alarms given by the system.

MPSP is formally described as a discrete optimization model.

The Multilevel Passenger Screening Problem (MPSP)

Parameters:

A set of N passengers, each characterized by an assessed threat value $AT_1, AT_2, \ldots, AT_N$ with $0 < AT_i \leq 1,\ i = 1, 2, \ldots, N$,

a set of M classes,

a set of V devices types, where device type k has capacity $c_k,\ k = 1, 2, \ldots, V$,

a subset of device types associated with each class, with $d_{ik} = 1(0)$ if class $i = 1, 2, ..., M$ uses (does not use) device type $k = 1, 2, ..., V$,

the *security level* of each class, L_i, $0 \leq L_i \leq 1,\ i = 1, 2, \ldots, M$.

Denote passenger assignments for the N passengers to the M classes by $A_1, A_2, \ldots, A_M$, where $A_i \subseteq \{1, 2, \ldots, N\}$ represents the subset of passengers who are assigned to class i. Define the *risk level* R_i of class $i = 1, 2, \ldots, M$ as the proportion of assessed threat values of the passengers assigned to class i (see also (2.1)). Therefore,

$$R_i = \left(\frac{1}{\sum_{j=1}^{N} AT_j} \right) \sum_{j=1}^{N} AT_j x_{ij},\ i = 1, 2, \ldots, M. \tag{3.1}$$

Find passenger assignments $A_1, A_2, \ldots, A_M$ such that $\bigcup_{i=1}^{M} A_i = \{1, 2, \ldots, N\}$ and $A_{i_1} \cap A_{i_2} = \emptyset$ for $i_1,\ i_2 = 1, 2, \ldots, M,\ i_1 \neq i_2$, and such that the device capacity constraints are satisfied (i.e.,

$\sum_{i=1}^{M} \sum_{j=1}^{M} d_{ik}|A_i| \leq c_k,\ k = 1, 2, \ldots, V$) and the total security is maximized (i.e., $\sum_{i=1}^{M} L_i R_i$).

MPSP is formulated as an integer program (IP) with binary decision variables, where $x_{ij} = 1(0)$ if passenger j is (not) assigned to class i for $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, N$. The IP formulation of MPSP is given by

$$z = \max \sum_{i=1}^{M} L_i R_i = \left(\frac{1}{\sum_{j=1}^{N} AT_j} \right) \sum_{i=1}^{M} \sum_{j=1}^{N} L_i AT_j x_{ij} \tag{3.2}$$

$$\text{subject to } \sum_{i=1}^{M} \sum_{j=1}^{N} d_{ik} x_{ij} \leq c_k,\ k = 1, 2, \ldots, V \tag{3.3}$$

$$\sum_{i=1}^{M} x_{ij} = 1,\ j = 1, 2, \ldots, N \tag{3.4}$$

$$x_{ij} \in \{0, 1\},\ i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, N. \tag{3.5}$$

The objective (3.2) is to maximize the total security, which is captured by the sum of the products of the security and risk levels. It can be rewritten as

$$z = \max \sum_{i=1}^{M} \sum_{j=1}^{N} p_{ij} x_{ij}$$

with $p_{ij} = L_i AT_j / (\sum_{k=1}^{N} AT_k)$ being the "profit" associated with assigning passenger j to class i. Constraint set (3.3) ensures that each device is operating within its capacity. Constraint set (3.4) ensures that each passenger is assigned to exactly one class. Constraint set (3.5) restricts x_{ij} to being 0-1 binary variables.

The linear programming relaxation (LP) of MPSP has objective (3.2) and constraints (3.3), (3.4), with (3.5) replaced by

$$x_{ij} \geq 0,\ i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, N. \tag{3.6}$$

Note that MPSP simultaneously assigns a set of N passengers to M classes, where the N passengers are expected to arrive at the airport in a given time period (based on flight departure times). This is reasonable since a very large fraction of passengers purchase their tickets at least several days prior to their departure times. Secure Flight uses information provided at the point

of ticket purchase to determine how a passenger is screened, and hence, the TSA has the ability to assign passengers to classes prior to passengers checking in for their flights [85]. Therefore, MPSP captures the scenario when passengers check in for their flights several hours before departure. Note that passenger arrival rates and device capacities are likely to change dramatically during a day due to the rate of flight departures and the number of security personnel scheduled to work at the airport. However, given that most passengers purchase their tickets in advance and security personnel schedules are determined in advance, then these parameters can be predicted with reasonable accuracy.

MPSP is motivated by MAP. In both MAP and MPSP, each passenger must be assigned to a class. A solution to MAP determines which classes should be used, indirectly determining the number of each device type to purchase, whereas a solution to MPSP determines how devices should be utilized once they are purchased and installed. In MAP, each class $i = 1, 2, \ldots, M$ is defined by its fixed cost FC_i (i.e., the cost to purchase and install devices associated with the class) and its marginal cost MC_i (i.e., the direct cost to screen a passenger with the class), with a budget B for screening the N passengers. Moreover, with MAP, passengers must be screened by classes such that the resulting assignment is budget feasible (i.e., $\sum_{i=1}^{M}\sum_{j=1}^{N} MC_i x_{ij} + \sum_{i=1}^{M} FC_i y_i \leq B$, where $y_i = 1$ if $\sum_{j=1}^{N} x_{ij} > 0$, $i = 1, 2, ..., M$). Computational experiments for MAP suggest that using two classes provides effective passenger screening [52]. The classes in MAP are implicitly defined by the device types that they use, whereas the classes in MPSP are explicitly defined by the device types that they use. For MPSP, each device type has a capacity that limits the number of passengers that can be screened in a given unit of time. Passengers are assigned such that each device type is capacity feasible.

The objective function, security levels, assessed threat values are interpreted in the same manner as in MAP (see Chapter 2). The total security is interpreted as the overall true alarm rate, and this is one of many possible interpretations of the total security. The total security can be used to capture alternative objective functions by redefining how the assessed threat values and security levels are defined.

MPSP does not explicitly incorporate stochastic screening times, passengers arriving in real-time, or space requirements of screening devices. Although screening time is an important operational factor, the time necessary for going through security procedures under normal operating conditions, including waiting time, is less than the time airlines recommend for passengers to ar-

rive before their flights depart (generally at least sixty minutes before departure time for domestic flights). Therefore, screening times are implicitly modeled by MPSP. Note also that MPSP does not consider passengers arriving in real-time. This is reasonable, since the TSA has access to information provided by passengers at the time of ticket purchase to determine the passengers' assessed threat values. In addition, screening devices may require significant amounts of space in airport lobbies or terminals. It is assumed that the classes are defined such that the necessary number of screening devices used by the classes is confined to the physical space available, and hence, space requirements are implicitly captured by MPSP.

MPSP is a particular case of the Collectively Capacitated Generalized Assignment Problem (CCGAP) [74, 75]. CCGAP generalizes MPSP by replacing the 0-1 weights d_{ik} with $a_{ijk} \in Z_0^+$, $i = 1, 2, \ldots, M,\ j = 1, 2, \ldots, N,\ k = 1, 2, \ldots, V$. CCGAP is only analyzed as a stochastic model, and hence, the algorithms and heuristics developed for CCGAP are not appropriate to be applied to MPSP. Toktas et al. [74] find approximate solutions for the deterministic formulation of CCGAP with uncertain resource capacities (i.e., c_k are not known). Toktas et al. [75] develop exact and approximate algorithms for the stochastic programming formulation of CCGAP with uncertain resource capacities.

3.2 Integer and Linear Programming Results

This section presents properties of MPSP. The first two results consider the discrete optimization formulation; they show that MPSP is NP-hard and show a property of the optimal solution to MPSP. The last two results consider the linear programming relaxation of MPSP; they provide an upper bound on the number of fractional variables in an LP solution and conditions under which the LP solution is integer.

First, consider the Multilevel Passenger Screening Feasibility Problem, which formulates MPSP as a decision problem in which the objective is to find a feasible solution to MPSP.

The Multilevel Passenger Screening Feasibility Problem (MPSFP)
Given:

$N,\ M,\ V,\ AT_1, AT_2, \ldots, AT_N,\ c_1, c_2, \ldots, c_V,\ d_{ik},\ i = 1, 2, ..., M,\ k = 1, 2, ..., V$ as given by MPSP.

Question: Denote the number of passengers assigned to the M classes by $n_1, n_2, \ldots, n_M$. Is there an assignment of passengers to classes $n_1, n_2, \ldots, n_M$ such that $\sum_{i=1}^{M} n_i = N$ and such that

each device type is operating within its capacity (i.e., $\sum_{i=1}^{M} d_{ik} n_i \leq c_k, k = 1, 2, \ldots, V$)?

Theorem 3.1 shows that MPSFP is NP-complete. This is shown by formulating a polynomial transformation from the Set Packing Problem, which is NP-complete [34]. First, the Set Packing Problem is formally stated.

Set Packing Problem
Given:

A finite set S,

a collection T of subsets of S,

a positive integer, $K \leq |T|$.

Question: Does T contain K mutually disjoint sets?

Theorem 3.1 *MPSFP is NP-complete.*

Proof. Given an arbitrary instance of the Set Packing Problem, a particular instance of MPSFP can be constructed in which the set of devices is equivalent to S (hence, $V = |S|$), $M = |T|$, $d_{ik} = 1(0)$ if $k \in T_i$, $i = 1, 2, ..., M$, $k = 1, 2, ..., V$, device capacities $c_k = 1$, $k = 1, 2, \ldots, V$, the number of passengers $N = K$, and $AT_1 = AT_2 = \ldots = AT_N = 1$. This transformation can be done in $O(|S| + |T| + K)$ time.

Suppose $n_1, n_2, \ldots, n_M$ is a solution to MPSFP. Note that $n_i \leq 1$, $i = 1, 2, ..., M$, since the device capacities are one. A solution to the Set Packing Problem, $\mathcal{T}$, can be constructed as follows: $T_i \in \mathcal{T}$ if $n_i > 0$, and $T_i \notin \mathcal{T}$ otherwise. Note that $|\mathcal{T}| = K$. Since the capacity constraints in MPSFP are satisfied (i.e., $\sum_{i:n_i>0} d_{ik} n_i \leq 1$ for all $n_i > 0$, $i = 1, 2, ..., M$), then $T_i \in \mathcal{T}$ are mutually disjoint.

Conversely, suppose $\mathcal{T} \subseteq T$ is a solution to the Set Packing Problem. A solution to MPSFP can be constructed as follows: $n_i = 1$ if $T_i \in \mathcal{T}$, and $n_i = 0$ otherwise, $i = 1, 2, \ldots, M$. Note that the assignment constraint is satisfied since all N passengers are assigned to a class since $|\mathcal{T}| = N$. Since all $T_i \in \mathcal{T}$ are mutually disjoint and $d_{ik} = 1(0)$ if $k \in T_i$, $i = 1, 2, ..., M$, then $n_1, n_2, \ldots, n_M$ obey the capacity constraints. □

MPSP is NP-hard since MPSFP is NP-complete. Note that the MPSP can be restricted and still remain NP-hard. In particular, MPSP remains NP-hard when the devices capacities are restricted

to one, and $L_i = 0$, $i = 1, 2, \ldots, M$, and for all security level values and risk level functions. There is likely to be a base class whose device capacities are sufficient for screening all passengers, and hence, a trivial solution to MPSFP would be available.

Note that Lemma 2.1 also applies to MPSP solutions, which relates the individual passenger assignments for to the total number of passengers assigned to a class. For any solution to MPSP, assigning passengers with higher assessed threat values to classes with higher security levels yields a higher objective function value. Without loss of generality, assume that $AT_1 \leq AT_2 \leq \ldots \leq AT_N$ and $L_1 \leq L_2 \leq \ldots \leq L_M$. Lemma 2.1 applies to both MAP and MPSP because each passenger is assigned to a class in a feasible solution, and hence, each assessed threat value is matched with a security level. Lemma 2.1 indicates that an optimal solution to MPSP determines $M - 1$ "breakpoints" in the assessed threat values, and all passengers with assessed threat values between two breakpoints are assigned to the same class.

In the LP relaxation, passenger $j = 1, 2, \ldots, N$ can be assigned to multiple classes with fractional assignments $x_{ij} > 0,\ i = 1, 2, \ldots, M$ such that $\sum_{i=1}^{M} x_{ij} = 1$. Consider an optimal solution to the LP relaxation. Let N_1 denote the passengers who have integer passenger assignments, and let N_2 denote the number of passengers whose assignments are split between two or more classes (i.e., $N_1 + N_2 = N$). Let V_1 denote the number of devices that are not at capacity, and let V_2 denote the number of devices at capacity (i.e., $V_1 + V_2 = V$). Theorem 3.2 shows that $N_2 \leq V_2$. Theorem 3.2 is a direct application of the main result given by Benders and van Nunen [7].

Theorem 3.2 *The number of passengers whose assignments are split between two or more classes in an optimal solution to the linear programming relaxation of MPSP is bounded above by the number of devices at capacity.*

Proof. The total number of variables in the LP relaxation is $NM + V$, and the total number of constraints is $N + V$. Let η be the average number of classes the split passengers are assigned to (i.e., $\eta \geq 2$). Therefore, the total number of nonzero variables is equal to

$$N_1 + N_2\eta + V_1 = N + (\eta - 1)N_2 + V_1.$$

Since there are $N + V$ constraints, then there are at most $N + V$ nonzero basic variables, and

$$N + (\eta - 1)N_2 + V_1 \leq N + V.$$

Then,

$$N_2 \leq \frac{V - V_1}{\eta - 1} = \frac{V_2}{\eta - 1} \leq V_2,$$

since $\eta \geq 2$. □

Theorem 3.2 provides an upper bound on the number of fractional variables in a solution to the LP relaxation of MPSP, $V_2 M$. Note that if the number of devices is much smaller than the number of passengers, then nearly all passenger assignments in an optimal solution to the LP relaxation are integer.

The assignment constraints (3.4) can be written as multiple-choice constraints in the IP formulation of MPSP. In this case, constraint set (3.4) is replaced by multiple-choice constraints

$$\sum_{i=1}^{M} x_{ij} \leq 1, \ j = 1, 2, \ldots, N.$$

The other constraints include (3.3) and (3.5). The objective function for the formulation with multiple-choice constraints becomes

$$z^{MC} = \sum_{i=1}^{M} \sum_{j=1}^{N} \hat{p}_{ij} x_{ij},$$

where $\hat{p}_{ij} = (L_i AT_j / \sum_{k=1}^{N} AT_k) + 1$ is the multiple-choice profit for assigning passenger $j = 1, 2, ..., N$ to class $i = 1, 2, \ldots, M$. This formulation of MPSP does not explicitly require each passenger to be assigned to a class, however, any assignment of all N passengers to classes has a larger objective function than any assignment of $N' < N$ passengers since $N \leq z^{MC} \leq N+1$ in any solution with all N passengers assigned to classes. Recall that z is the optimal objective function value to the original formulation of MPSP. Then $z = z^{MC} - N > 0$ only if the multiple-choice constraints are satisfied.

Under certain conditions, the LP relaxation of MPSP provides integer passenger assignments for all choices of the device capacities, assessed threat values, and security level values. To show these conditions, consider the LP relaxation of the formulation of MPSP with multiple-choice constraints. The constraints to this problem can be written in the form $Ax \leq b$, where A, x and b are matrices. Let $x = (x_{1,1}, x_{1,2}, \ldots, x_{1,M}, x_{2,1}, \ldots, x_{M,N})^T$, and $b = (c_1, c_2, \ldots, c_V, 1^N)^T$, where 1^N denotes a

row vector of N ones. Then, A is a $(V+N) \times (M\,N)$ matrix of the form

$$A = \begin{bmatrix} \Lambda_1 & \Lambda_2 & \cdots & \Lambda_M \\ I & I & \cdots & I \end{bmatrix} = \begin{bmatrix} A_C \\ A_{MC} \end{bmatrix}.$$

Each $\Lambda_i,\ i = 1, 2, \ldots, M$ is defined as a $V \times N$ matrix such that $\Lambda_i = [d^i\ d^i \cdots d^i]$ (i.e., a column vector repeated N times), where $d^i = (d_{i1}, d_{i2}, \ldots, d_{iV})^T$.

The optimal solution to the LP relaxation of MPSP contains integer values for all security level values, assessed threat values, and device capacities if A is totally unimodular (TU). Theorem 3.3 describes conditions under which A is TU.

Theorem 3.3 *The following two statements are equivalent:*
(i) A is TU,
(ii) For every subset of devices $K \subseteq \{1, 2, \ldots, V\}$, there exists a partition, K_1 and K_2 of K, such that

$$0 \leq \sum_{k \in K_1} d_{ik} - \sum_{k \in K_2} d_{ik} \leq 1,\ i = 1, 2, \ldots, M.$$

Proof. Note that A is TU if and only if A^T is TU. The claim is shown using Theorem 2.7 from Nemhauser and Wolsey [59] applied to A^T.

Theorem (Nemhauser and Wolsey) *The following statements are equivalent:*
(a) A is TU, and
(b) For every $Q \subseteq \{1, 2, \ldots, V+N\}$, there exists a partition, Q_1 and Q_2 of Q, such that

$$\left| \sum_{i \in Q_1} a_{ij} - \sum_{i \in Q_2} a_{ij} \right| \leq 1,\ j = 1, 2, \ldots, N \times M,$$

where a_{ij} is an element of A.

It is sufficient to show that statements *(ii)* and *(b)* are equivalent. First, it is shown that *(ii)* $\Rightarrow$ *(b)*. First, consider the case when $K = \emptyset$, and $Q \subseteq \{V+1, V+2, \ldots, V+N\}$ (i.e., a subset of the multiple-choice constraints). Then *(b)* trivially holds since each column in A_{MC} is composed of $N-1$ zeros and a single one.

Consider the case when $K \neq \emptyset$ and $Q = K$ (i.e., a subset of the capacity constraints). If *(ii)* holds for $K_1,\ K_2$, then *(b)* holds for $Q_1 = K_1$ and $Q_2 = K_2$ since A_C is composed of M distinct columns (i.e., $d^1, d^2, \ldots, d^M$) that are each copied N times. Finally, consider the case when $K \neq \emptyset$

and $Q = K \cup Q'$, where $Q' \subseteq \{V+1, V+2, \ldots, V+N\}$. If *(ii)* holds for $K_1,\ K_2$, then *(b)* holds for $Q_1 = K_1$ and $Q_2 = K_2 \cup Q'$.

To show that *(b)* $\Rightarrow$ *(ii)*, let *(b)* hold for all subsets $Q = Q' \cup Q''$, where $Q' \subseteq \{1, 2, \ldots, V\}$ (i.e., a subset of the capacity constraints) and $Q'' \subseteq \{V+1, V+2, \ldots, V+N\}$ (i.e., a subset of the multiple-choice constraints). Since each row in A_{MC} has M ones equally spaced by $N-1$ zeros, when a row of A_{MC} is added to Q'', then a one is added to a single column in each subset of rows of $\Lambda_1, \Lambda_2, \ldots, \Lambda_M$ (i.e., a subset of the rows of A_C given by Q') and zeros are added to the remaining columns. Recall that each Λ_i is composed of N identical vectors, d^i, $i = 1, 2, \ldots, M$. Then for a given Q' and all choices of Q'', there exists a partition, Q_1, and Q_2 of Q', such that

$$\left| \sum_{k \in Q_1} d_{ik} - \sum_{k \in Q_2} d_{ik} \right| \leq 1,\ i = 1, 2, \ldots, M$$

and

$$\left| \sum_{k \in Q_1} d_{ik} - \sum_{k \in Q_2} d_{ik} - 1 \right| \leq 1,\ i = 1, 2, \ldots, M.$$

Since both of these statements must be true simultaneously, then

$$0 \leq \sum_{k \in Q_1} d_{ik} - \sum_{k \in Q_2} d_{ik} \leq 1,\ i = 1, 2, \ldots, M.$$

Therefore, *(ii)* holds for $K_1 = Q_1$ and $K_2 = Q_2$. $\square$.

First, consider the case with two device types. There are two conditions under which the optimal solution to the LP relaxation of MPSP is integer: when each class uses no more than one device type or when every class that uses the second device type also uses the first device type.

The conditions in Theorem 3.3 are restrictive and not likely to hold in practice. For general values of V, the optimal solution to the LP relaxation of MPSP is integer when there is no overlap of the device types used by each of the classes (i.e., the subsets of devices used by the classes are disjoint), or when the classes can be sorted such that class $i = 1, 2, \ldots, M-1$ uses a subset of devices that class $i' = i+1, i+2, \ldots, M$ uses.

3.3 Computational Results

This section reports computational results for MPSP using an example that incorporates flight schedule and passenger volume data extracted from the Official Airline Guide (OAG) for the domestic flights of a single airline carrier at distinct airport stations in the United States. The data provided by the OAG includes the set of flights, the number of available seats on each flight, and the departure time of each flight [60].

To obtain the number of passengers, the peak number of passengers in any ten minute interval is estimated using flight data extracted from the OAG. The flights (and the corresponding flight numbers) departing from a single terminal at a hub airport are considered. Each of the flight numbers is scheduled to depart from the terminal with a given frequency, with some flight numbers operating daily and others operating infrequently. Since including these infrequent flights would inaccurately increase the peak number of passengers, only flight numbers operating with a frequency of at least three flights per week (twelve per month) are considered. To find the peak number of passengers in a ten minute time window, passengers are assumed to arrive randomly according to a uniform distribution between 30 and 90 minutes prior to the departure time of each flight. This arrival interval is recommended by airlines for domestic flights, with 30 minutes prior to departure being the latest check-in time (e.g., see www.nwa.com, www.ual.com, www.aa.com, or www.delta.com). Moreover, each flight is assumed to have an enplanement rate of 80% (i.e., the number of passengers divided by the number of available seats). The passenger arrival rates are super-positioned for each of the flights, and the data set is chosen based on finding the largest expected number of arriving passengers in a 10 minute window, resulting in $N = 916$ passengers. In order to consider the sensitivity with respect to passenger volume, ten passenger volumes are considered: 10%, 20%,..., 100% of the peak passenger load of 916, rounded to the nearest whole number.

Four devices are considered, two for screening checked baggage, and two for screening passengers and their carry-on baggage. These device types are chosen since Secure Flight is a binary system, and hence, there are two ways to screen passengers and their baggage. The chosen device types are based on current devices and procedures used to screen passengers in commercial airports in the United States. The two devices for screening passengers and carry-on baggage are assumed to be a combination of metal detectors and X-ray machines, labeled D1, and a combination of trace portals and X-ray machines, labeled D2. The two devices for screening checked baggage

are explosive detection systems (EDSs), labeled D3, and explosive trace devices, labeled D4. The combination of metal detectors (trace portals) and X-ray machines is considered as one device for two reasons. First, passengers and carry-on baggage are screened together, with passengers always in the presence of their carry-on baggage. For this reason, either the metal detector (trace portal) or the X-ray machine is the screening bottleneck, as passengers must either wait for their carry-on baggage screening to be completed or carry-on baggage must wait for passenger screening to complete. Therefore, the capacity (i.e., throughput) of these two devices is the same. The second reason is that these devices work together to remove threat items. Once the passengers complete the screening procedures, threat passengers have access to threat items in carry-on baggage after screening, whereas they do not have access to threat items in checked baggage.

Table 3.1 summarizes the false clear rates and capacities associated with the devices. The device values are estimated using information available in the public domain [12, 89, 52]. It is assumed that the devices for passengers operate independently of the devices for checked-baggage screening. This is reasonable since the screening for these two classes of devices is often done in different areas in airports, and the system response for checked baggage (i.e., either alarm or clear) does not depend on the system response for passengers and carry-on baggage.

Table 3.1: MPSP Security Device Data

Label	Device Type	False Clear	Units/hour	Units/10 minutes
D1	Metal Detector/X-ray Machine	0.20	120	20
D2	Trace Portal/X-ray Machine	0.15	90	15
D3	EDS	0.12	150	25
D4	Explosive Trace Devices	0.15	30	5

Nine classes are defined as follows. Passengers and carry-on baggage can be screened in three ways: by D1, by D2, or by both D1 and D2. Likewise, checked baggage can be screened in three ways: by D3, by D4, or by both D3 and D4. The nine classes enumerate all possible ways passengers and baggage can be screened given that each are screened by at least one device type. The security level of the classes represent the average of the true alarm rates associated with passenger screening and checked-baggage screening. Note that a different metric could be used to determine the security level of a class based on the false clear rates associated with both types of devices. Without loss of generality, assume that passengers (checked baggage) are screened by D1 before D2 (D3 before D4) when screened by two devices.

When passengers or checked baggage are screened by both devices, it is assumed that the system gives an alarm response if the passenger or bag triggers an alarm at either device in the system. Therefore, the system gives a clear response only if both devices give a clear response. First, define T as the event that a passenger is a threat, and Ci as the event that a passenger is cleared by device Di, $i = 1, 2, 3, 4$. Kobza and Jacobson [38] define the dependence between two screening devices for passengers as

$$\varepsilon^{T}_{C2|C1} = P(C2|C1 \cap T) - P(C2|T), \tag{3.7}$$

with the lower and upper bounds for $\varepsilon^{T}_{C2|C1}$ being

$$-P(C2|T) \leq \varepsilon^{T}_{C2|C1} \leq \min\left(P(C2|C1 \cap T),\ \frac{P(C2|T)P(\bar{C}1|T)}{P(C1|T)}\right). \tag{3.8}$$

Then, the false clear rate when being screened by both devices is

$$P_{FC}^{D1,D2} = P_{FC}^{D1}(P_{FC}^{D2} + \varepsilon^{T}_{C2|C1}). \tag{3.9}$$

Devices D1 and D2 operate independently if $\varepsilon^{T}_{C2|C1} = 0$. If knowing that a threat is cleared by device D1 increases (decreases) the probability of a false clear by device D2, then there is a positive (negative) dependence and $\varepsilon^{T}_{C2|C1} > (<) 0$.

The dependence between the two screening devices for checked baggage $\varepsilon^{T}_{C4|C3}$ is defined by substituting D3 for D1 and D4 for D2 in (3.7), (3.8), and (3.9). Based on the bounds given in (3.8), $\varepsilon^{T}_{C2|C1} = \varepsilon^{T}_{C4|C3} = 0.1$ for the scenarios considered. This results in a conservative relationship between the screening devices in which the false clear rate associated with the combination of device types is greater than when the device types are assumed to operate independently. The security level is computed as the true alarm rate, $1 - P_{FC}^{D1,D2}$.

Table 3.2 contains class information, the devices used by each of the classes and the security level associated with each class. The classes are sorted such that $L_1 < L_2 < \ldots < L_9$.

The example considers four assessed threat value distributions. Initially, all passengers are assumed to be identical, resulting in a degenerate distribution with all assessed values equal to one (Type I). This corresponds to the case when no information is known about the passengers.

The remaining assessed threat values assume that most passengers can be cleared of being a threat and hence, have low assessed threat values. This is consistent with what is known in the public domain about how Secure Flight works [78]. The second assessed threat distribution is an

Table 3.2: MPSP Class data

	d_{ik}				
Class (i)	$k=1$	$k=2$	$k=3$	$k=4$	Security Level L_i
1	1	0	0	1	0.825
2	1	0	1	0	0.84
3	0	1	0	1	0.85
4	0	1	1	0	0.865
5	1	0	1	1	0.885
6	1	1	0	1	0.90
7	0	1	1	1	0.91
8	1	1	1	0	0.915
9	1	1	1	1	0.96

exponential random variate with mean 1/16, where values greater than one are discarded when assessed threat values are randomly generated (labeled Type III). A randomly generated Type III assessed threat value is greater than one with a probability of approximately 10^{-7}. The third type of assessed threat distribution is a triangular distribution with probability density function

$$f(W_j) = 2(1 - AT_j), \tag{3.10}$$

for $0 \leq AT_j \leq 1$ (labeled Type IV). The fourth assessed threat distribution has two parts: a triangular distribution for assessed threat values less than 0.1 and a uniform distribution for assessed threat values between 0.1 and 1.0. The probability density function is

$$f(AT_j) = \begin{cases} (341 - 3400AT_j)/18, & 0 \leq AT_j < 0.1 \\ 1/18, & 0.1 \leq AT_j \leq 1.0 \end{cases} \tag{3.11}$$

$j = 1, 2, ..., N$ (labeled Type V). This assessed threat distribution results in 95% of the assessed threat values being less than 0.1.

The example has 40 scenarios consisting of all combinations of ten passenger volumes and four assessed threat distributions. The 40 scenarios were formulated as LPs (with objective (3.2) and constraints (3.3), (3.4), (3.6)). The LPs were solved using Matlab on a 2.6 GHz Pentium IV processor with 1.0 GB of RAM. All the LPs were solved in fewer than four seconds. All of the LPs were integer, and hence, the scenarios did not need to be formulated as IPs.

For each of the 40 scenarios, the objective function values to the optimal IP solution, z, for its

four scenarios have the following property:

$$z^{\text{Type I}} < z^{\text{Type IV}} < z^{\text{Type III}} < z^{\text{Type V}}.$$

Note that the Types IV, III, V assessed threat distributions result in approximately 20%, 80%, and 95% of the passengers having assessed threat values less than 0.1, respectively. These assessed threat distributions model situations when relatively few passengers are perceived to be potential risks, hence most passengers have low assessed threat values. This suggests that if the prescreening system has the ability to assign low assessed threat values to the majority of passengers, then the resulting multilevel screening system may be more effective in preventing attacks.

To describe the results, the passenger *partition* denotes the number of passengers assigned to each class in an IP solution, $n_1, n_2, \ldots, n_M$. Lemma 2.1 show how the optimal passenger assignments can be determined given the partition to an optimal solution, and hence, the passenger partition is sufficient for describing the optimal solution.

Table 3.3 summarizes the partitions for the 40 scenarios. Given the passenger volume and assessed threat distribution, it shows the optimal partition and the number of devices at capacity D' (i.e., the number of passengers screened by the device is equal to its capacity). Note that the passenger partitions is the same across all four assessed threat values except when $N = 916$, which indicates that the optimal solutions are not sensitive to the assessed threat distributions for this example. No more than six classes are used, and the number of devices at capacity is always at least one less than the number of classes in use. When $N = 916$, being able to differentiate passengers results in five classes being used, as opposed to six being used when passengers are not differentiated. This suggests that fewer classes may be desirable when the prescreening system is more accurate in identifying passenger risk. Note that all of the available devices are used when the passenger volume is 90% or greater. All of the available D2 and D4 devices are used when the passenger volume is 20% or greater. This suggests that scarcer device types (i.e., devices whose capacities are much less than the number of passengers) are at or near capacity even when there are few passengers to screen.

The sensitivity of the solutions is studied in two ways. The first approach studies the sensitivity with respect to the device capacities. In this case, the capacity of device type k is increased to $c_k + \Delta_c$, with $\Delta_c = 0.05c_k,\ 0.10c_k$, $k = 1, 2, 3, 4$, resulting in eight perturbed scenarios. The second approach studies the sensitivity with respect to the number of passengers. In this case, $N + \Delta$

Table 3.3: Optimal IP Solutions for Type I, III, IV, V Assessed Threat Distributions

			Optimal Partition									
N	Volume	AT Type	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	D'
916	100%	I	5	731	14	0	0	0	72	82	12	4
916	100%	III, IV, V	91	645	0	86	0	0	10	0	84	4
824	90%	I, III, IV, V	0	639	0	0	5	0	4	0	176	3
733	80%	I, III, IV, V	0	548	0	0	5	0	0	0	180	2
641	70%	I, III, IV, V	0	456	0	0	5	0	0	0	180	2
550	60%	I, III, IV, V	0	365	0	0	5	0	0	0	180	2
458	50%	I, III, IV, V	0	273	0	0	5	0	0	0	180	2
366	40%	I, III, IV, V	0	181	0	0	5	0	0	0	180	2
275	30%	I, III, IV, V	0	90	0	0	5	0	0	0	180	2
183	20%	I, III, IV, V	0	0	0	0	3	0	0	0	180	1
92	10%	I, III, IV, V	0	0	0	0	0	0	0	0	92	0

passengers are considered, with $\Delta = \pm 0.05N, \pm 0.1N$. The scenarios were reformulated as LPs and solved using Matlab.

The perturbed passenger partitions are compared to the original passenger partition for each of the 40 scenarios. In many cases, the perturbed passenger partition is nearly identical to the original, where several of the passengers in the perturbed solution are reallocated to one of the existing classes in the original solution such that the assignment and capacity constraints are obeyed. To describe these solutions, let I_0 denote the subset of classes to which at least one passenger is assigned in the optimal solution to the original scenario. Let I_P denote the subset of classes to which at least one passenger is assigned in the optimal solution to the perturbed scenario. A perturbed partition is labeled *similar* if $I_P \subseteq I_0$ and *dissimilar* otherwise.

In most cases, the perturbed partitions are similar to the original partitions, suggesting that the changes in capacity and passenger volume do not prescribe additional classes to be used. The perturbed partitions are identical to the original partitions when the capacities of devices not at capacity are increased.

Table 3.4 summarizes the sensitivity results. Note that the results are summarized across the assessed threat distributions when possible. The Type I scenario with 100% of the passenger volume has the most number of dissimilar solutions, which suggests that being able to differentiate passengers results in less sensitive solutions. Note that when the passenger volume is 100%, the $\Delta_c = 0.05c_3$ scenario is dissimilar for the Type I distribution. This is counterintuitive since D3 is a more abundant resource and results in identical passenger partitions when $N < 916$. Note that

Table 3.4: MPSP Sensitivity Results

N	AT Type	Dissimilar Partitions, Passenger Volume	Dissimilar Partitions, Device Capacities
916	I	$\Delta = \pm 0.05N, 0.1N$	$\Delta_c = 0.1c_1, 0.05c_2, 0.1c_2, 0.05c_3$
916	III, IV, V	$\Delta = \pm 0.1N$	$\Delta_c = 0.05c_1, 0.1c_1, 0.1c_2$
824	I, III, IV, V	$\Delta = 0.05N, 0.1N$	$\Delta_c = 0.05c_2, 0.1c_2$
733	I, III, IV, V		$\Delta_c = 0.05c_2, 0.1c_2$
641	I, III, IV, V		$\Delta_c = 0.05c_2, 0.1c_2$
550	I, III, IV, V		$\Delta_c = 0.05c_2, 0.1c_2$
458	I, III, IV, V		$\Delta_c = 0.05c_2, 0.1c_2$
366	I, III, IV, V		$\Delta_c = 0.05c_2, 0.1c_2$
275	I, III, IV, V		$\Delta_c = 0.05c_2, 0.1c_2$
183	I, III, IV, V	$\Delta = 0.05N, 0.1N$	$\Delta_c = 0.05c_2, 0.1c_2$

there are no dissimilar solutions with respect to changes in passenger volume when the passenger volume is between 30% and 90%.

3.4 Conclusions

This chapter introduces the Multilevel Passenger Screening Problem that models passenger and baggage screening systems. MPSP models each security class in terms of its security level and the devices it uses, and passenger assignment and device capacity constraints must be satisfied. MPSP is formulated as an integer programming model. Data extracted from the OAG was used to construct the passenger set for 40 scenarios for MPSP across four assessed threat distributions. The optimal LP solutions were computed for each of these examples. The results suggest that the screening system is more effective when more passengers have low assessed threat values. Moreover, the optimal LP solutions are integer for all 40 scenarios. Studying the sensitivity of the solutions indicates that passengers are often not assigned to new classes when there are changes in the device capacities or passenger volume.

Chapter 4

The Sequential Stochastic Multilevel Passenger Screening Problem

This chapter extends Multilevel Passenger Screening Problem (MPSP) [51] to consider the real-time operation of passenger screening systems. Understanding how real-time passenger screening systems operate is important, since many commercial airports have been burdened with long security lines for passenger screening as a result of widespread operational changes at the nation's airports [20]. Developing improved screening methods, allocating and using security screening devices, and keeping passenger waiting and screening times short, has proven to be quite challenging.

The Sequential Stochastic Multilevel Passenger Screening Problem (SSMPSP) extends MPSP that models multilevel passenger screening strategies using Markov Decision Processes and discrete optimization models. In MAP and MPSP, the set of passengers to be screened at a particular station in an airport in a given period of time is assumed to be known, and hence, the assessed threat values are assumed to be known a priori. This assumption is relaxed by SSMPSP, in which passengers check in sequentially, and each passenger's assessed threat value becomes known upon check-in. This necessitates a change in the solution methodology since passenger screening decisions are made simultaneously in MAP and MPSP, whereas a series of passenger screening decisions are made sequentially for SSMPSP.

The primary contribution of this chapter is to identify a real-time methodology for screening passengers in a multilevel screening paradigm and to show how the methodology can be used to provide insights into the operation and performance of such real-time systems. Note that this research assumes that a prescreening system such as Secure Flight has been implemented and is effective in identifying passenger risk (i.e., the assessed threat values accurately quantify passenger risk).

This chapter is organized as follows. Section 4.1 introduces SSMPSP as a Markov Decision Process, and shows how the optimal policy for SSMPSP can be obtained by dynamic programming. Section 4.2 describes the Sequential Stochastic Assignment Heuristic (SSA) for SSMPSP. SSA

defines a policy for assigning passengers to security resources. A property is provided under which SSA obtains the optimal policy. Section 4.3 provides examples using passenger volume and flight data extracted from the Official Airline Guide [60] and discusses computational results for several scenarios based on these examples. The policy provided by SSA is analyzed to provide insight into the operation of real-time passenger screening systems. Section 4.4 provides concluding comments.

4.1 The Sequential Stochastic Multilevel Passenger Screening Problem

The Sequential Stochastic Multilevel Passenger Screening Problem (SSMPSP) generalizes MPSP to consider stochastic passenger arrivals. Both MPSP and SSMPSP consider N passengers who arrive at the airport to be screened in a given period of time. The primary difference between the two problems is that MPSP assigns N passengers to M classes simultaneously prior to the passengers checking in, while for SSMPSP, passengers check in sequentially, and each passenger is assigned to a class upon check-in (i.e., before the next passenger checks in). Note that the passengers may check in several hours before they arrive at the airport. Therefore, the time period when passengers are assigned to classes may be different than the time period when passengers arrive at the airport. Note that a prescreening system such as Secure Flight determines which passengers are selectees using information provided by passengers at the point of ticket purchase, and hence, assigning passengers to classes when they check in is a reasonable assumption [85]. Note that any instance of MPSP can be formulated as an instance of SSMPSP by arbitrarily ordering the N assessed threat values and by assuming that the assessed threat values are unknown until the passengers check in. To differentiate the passengers in the static and stochastic models, the N passengers are indexed as $j = 1, 2, \ldots, N$ in MPSP (when passenger order is not a factor), and as $t = 1, 2, \ldots, N$ in SSMPSP (when passenger order is critical).

Several assumptions are needed to define SSMPSP. First, N passengers arrive at the airport to undergo screening during a given time period. The number of passengers is known, but the individual passengers (and their assessed threat values) are unknown prior to check-in. Each passenger has exactly one checked and one carry-on bag. The device capacities represent the number of passengers or bags that a device can screen in the time period. It is assumed that each passenger's assessed threat value becomes known upon check-in. Let $\mathcal{AT}_1, \mathcal{AT}_2, \ldots, \mathcal{AT}_N$ denote the unknown assessed threat value of each passenger prior to check-in, which are identically and

independently distributed random variables. Let at_t denote the realized assessed threat value after passenger t checks in. The probability density function of the unknown assessed threat values is $f_{\mathcal{AT}_t}(at_t)$, which is identical for all passengers $t = 1, 2, \ldots, N$.

SSMPSP is formulated as a stochastic optimization problem, where the objective is to determine the optimal policy for assigning passengers to classes as they check in for their flights. A policy π defines a rule for assigning each passenger to a class, which may change after each passenger assignment is made. A policy may be deterministic (i.e., the policy always assigns a passenger to the same class under identical conditions) or random (i.e., the policy may assign a passenger to different classes under identical conditions). It may also be Markovian (i.e., the policy only depends on the current passenger and current conditions) or history-dependent (i.e., the policy depends on the passenger assignments of the previous passengers).

The Sequential Stochastic Multilevel Passenger Screening Problem (SSMPSP)

Given:

The number of passengers expected to undergo screening in a given time period N,

a set of M classes,

a set of V devices types, where device type k has capacity c_k, $k = 1, 2, \ldots, V$,

a subset of device types associated with each class, with $d_{ik} = 1(0)$ if class $i = 1, 2, ..., M$ uses (does not use) device type $k = 1, 2, ..., V$,

the *security level* of each class, L_i, $0 \leq L_i \leq 1$, $i = 1, 2, \ldots, M$,

a probability density function $f_{\mathcal{AT}_t}(at_t)$ that describes the distribution of the assessed threat values for passenger $t = 1, 2, \ldots, N$.

Objective: Denote passenger assignments for the N passengers to the M classes by $x_1, x_2, \ldots, x_N$, where $x_t \in \{1, 2, \ldots, M\}$, $t = 1, 2, \ldots, N$, represents the class to which passenger t is assigned. Find the policy π^* that determines the passenger assignments $x_1^{\pi*}, x_2^{\pi*}, \ldots, x_N^{\pi*}$ such that each device type is operating within its capacity (i.e., $\sum_{i=1}^{M} d_{ik} |\{t \ : \ x_t^{\pi*} = i\}| \leq c_k, \ k = 1, 2, \ldots, V$) and the expected total security (i.e., $E\{\sum_{t=1}^{N} L_i \mathcal{AT}_t | x_t^{\pi*} = i\}$) is maximized.

Since SSMPSP has a sequential structure, it is natural to formulate SSMPSP as a Markov Decision Process (MDP). This MDP formulation also illustrates how the optimal policy is found.

SSMPSP can be formulated as an MDP with $N+1$ stages using post-decision state variables, with the state in stage $t = 0, 1, \ldots, N$ describing the system after the first t passengers have been assigned to classes [68], where $t = 0$ corresponds to the initial stage.

Let S denote the set of states. Let state $\mathbf{s_t} \in S$ represent the vector of remaining capacities after the first $t = 0, 1, \ldots, N$ passengers have been assigned to classes, where $\mathbf{s_t} = (\vec{c}_1^t, \vec{c}_2^t, \ldots, \vec{c}_V^t)$, with $\vec{c}_k^t$ denoting the remaining capacity of device $k = 1, 2, \ldots, V$ after stage t. The initial state corresponds to the initial capacity, $\mathbf{s_0} = (c_1, c_2, \ldots, c_V)$. Therefore, $|S| = (c_1 + 1) \times (c_2 + 1) \times \cdots \times (c_V + 1)$. Moreover, define a vector $\mathbf{d^i} = (d_{i,1}, d_{i,2}, \ldots, d_{i,V})$ associated with class $i = 1, 2, \ldots, M$ that represents the vector of devices used by each class.

The set of actions available for assigning passenger t to a class is given by the subset of classes to which a passenger can be assigned. Given state $\mathbf{s_{t-1}}$, the set of available classes in stage t is given by $X_t(\mathbf{s_{t-1}}) \subseteq \{1, 2, \ldots, M\},\ t = 1, 2, \ldots, N$, for all $\mathbf{s_{t-1}} \in S$, with $i \in X_t(\mathbf{s_{t-1}})$ if $\vec{c}_k^t - d_{ik} \geq 0,\ k = 1, 2, \ldots, V$. Let $x_t \in X_t(\mathbf{s_{t-1}})$ denote the security class to which passenger t is assigned. The transition probabilities $p(\mathbf{s_t}|\mathbf{s_{t-1}}, x_t),\ t = 1, 2, \ldots, N$, determine state $\mathbf{s_t}$ given $\mathbf{s_{t-1}}$ and x_t. To define these probabilities for a deterministic policy, as each passenger is assigned to a class, the remaining capacities of the devices associated with the class assigned decrease by one, with

$$p(\mathbf{s_t}|\mathbf{s_{t-1}}, x_t) = \begin{cases} 1, \text{ if } \mathbf{s_t} = \mathbf{s_{t-1}} - \mathbf{d}^{x_t} \\ 0 \text{ otherwise} \end{cases}$$

for $t = 1, 2, \ldots, N$.

The objective function value of SSMPSP is determined by accruing a reward after each stage in the MDP. Define $r(\mathbf{s_{t-1}}, \mathcal{AT}_t, x_t)$ as the reward for assigning passenger t to class x_t given state $\mathbf{s_{t-1}} \in S$, $t = 1, 2, \ldots, N$,

$$r(\mathbf{s_{t-1}}, \mathcal{AT}_t, x_t) = L_{x_t}\mathcal{AT}_t,\ t = 1, 2, \ldots, N.$$

Given that passenger t has assessed threat value at_t, then the reward for assigning passenger t to class x_t becomes $L_{x_t}at_t$ after passenger t checks in. The reward in stage t corresponds to the amount of security obtained from screening passenger $t = 1, 2, \ldots, N$. If $X_t(\mathbf{s_{t-1}}) = \emptyset$ (i.e., there are no feasible passenger assignments), then the reward is $-N$. This results in a non-negative objective function value only if all passengers are assigned to classes, and hence, any large negative number whose absolute value is greater than N would also result in a non-negative objective function value.

The *expected total security* for SSMPSP is determined by policy π, which describes the decision rule for selecting an action (i.e., the class to which each passenger is assigned) in each state and at each stage. Let S_{t-1} represent the random variable corresponding to the state after $t-1$ passengers have been assigned to classes, and let $X_t^\pi(S_{t-1})$ represent the random variable corresponding to the class to which passenger t is assigned given policy π. Given that the system is initialized in state $\mathbf{s_0}$, then the expected total security for SSMPSP is defined as

$$E^\pi(\mathbf{s_0}) = E^\pi \left\{ \sum_{t=1}^{N} r(S_{t-1}, \mathcal{AT}_t, X_t^\pi(S_{t-1})) \,|S_0 = \mathbf{s_0} \right\},$$

where $r(S_{t-1}, \mathcal{AT}_t, X_t^\pi(S_{t-1}))$ is the random variable corresponding to the reward obtained in time period t in state S_{t-1} with decision $X_t^\pi(S_{t-1})$ based on policy π. The objective of SSMPSP is to find the optimal policy π^* such that $E^{\pi^*}(\mathbf{s_0}) = \max_\pi\{E^\pi(\mathbf{s_0})\}$. This optimal policy is a deterministic Markov policy since the number of states is finite [69].

Define the value function in stage $t = 1, 2, \ldots, N$ as the optimal expected total security for assigning the remaining $N - t$ passengers,

$$V_{t-1}(\mathbf{s_{t-1}}) = \mathrm{E}\left[\max_{x_t \in X_t(\mathbf{s_{t-1}})} \{r(\mathbf{s_{t-1}}, \mathcal{AT}_t, x_t) + V_t(S_t(x_t))\} \right],$$

for $t = 1, 2, \ldots, N$, where $S_t(x_t)$ denotes the state after passenger t has been assigned to class x_t. For SSMPSP, this is rewritten as

$$V_{t-1}(\mathbf{s_{t-1}}) = \mathrm{E}\left[\max_{x_t \in X_t(\mathbf{s_{t-1}})} \{L_{x_t}\mathcal{AT}_t + V_t(\mathbf{s_{t-1}} - \mathbf{d}^{x_t})\} \right], \tag{4.1}$$

for $t = 1, 2, \ldots, N$, with boundary conditions

$$V_N(\mathbf{s_N}) = 0, \ \mathbf{s_N} \in S, \tag{4.2}$$

which are also known as the *optimality equations*. The optimal policy π^*, which solves the optimality equations, can be found using dynamic programming.

The total security of a SSMPSP instance given realized assessed threat values at_1,at_2, ...,at_N and their assignments $x_1, x_2, \ldots, x_N$, is $\sum_{t=1}^{N} at_t L_{x_t}$. The total security can be rescaled to be

between zero and one, with

$$z^{SS} = \frac{\sum_{t=1}^{N} at_t L_{x_t}}{\sum_{t=1}^{N} at_t},$$

which can be directly compared to (3.2), the total security of MPSP. Note that the total security for MPSP is normalized by a factor of $\sum_{j=1}^{N} at_j$, a constant parameter. However, this value is not known for SSMPSP until after the N passengers have arrived.

4.2 The Sequential Stochastic Assignment Heuristic

Finding the optimal solution to the SSMPSP optimality equations (4.1) and (4.2) using a dynamic programming is computationally intractable. The optimality equations can only be solved in a reasonable amount of computation time for small instances of SSMPSP. In order to solve large instances of SSMPSP in real-time, the Sequential Stochastic Assignment Heuristic (SSA) is presented to efficiently obtain approximate solutions to SSMPSP.

The policy defined by SSA assigns passengers to classes by using the main result from Derman et al. [18]. Applying this result requires that the classes be sorted such that $L_1 \leq L_2 \leq \ldots \leq L_M$. There are two phases to this heuristic: the preprocessing phase and the assignment phase (see Algorithm 2 pseudocode). In the preprocessing phase, the number of passengers to assign to each class is determined for the passengers who are expected to arrive in the time period. A series of intervals are created to construct the expected value of the assessed threat values of the N passengers arriving in the time period as follows. Given the value of N and the probability density function of the assessed threat values $f_{\mathcal{AT}_t}(at_t)$ (with cumulative density function $F_{\mathcal{AT}_t}(at_t)$), a series of $t' = N - t + 1$ assignment intervals are constructed for passenger $t = 1, 2, \ldots, N$, where t' represents the number of passengers that have not yet been assigned to a class. Interval $j = 1, 2, \ldots, t'$ is defined by the boundaries $J_{t',j-1}$ and $J_{t',j}$, with $J_{t',0} \leq J_{t',1} \leq \ldots \leq J_{t',t'}$. These intervals can be determined by the recursion

$$J_{t'+1,j} = J_{t',j-1} F_{\mathcal{AT}_t}(J_{t',j-1}) + J_{t',j}[1 - F_{\mathcal{AT}_t}(J_{t',j})] + \int_{J_{t',j-1}}^{J_{t',j}} y \, dF_{\mathcal{AT}_t}(y), \tag{4.3}$$

with boundary condition $J_{t',0} = 0$ and $J_{t',t'} = 1$, $t' = 1, 2, \ldots, N$.

Define $E[AT_i]$ as the expected value of the ith smallest assessed threat value, $i = 1, 2, ..., N$, as

$$E[AT_j] = J_{N,j-1} F_{\mathcal{AT}_N}(J_{N,j-1}) + J_{N,j}[1 - F_{\mathcal{AT}_N}(J_{N,j})] + \int_{J_{N,j-1}}^{J_{N,j}} y \, dF_{\mathcal{AT}_N}(y),$$

[18]. The expected assessed threat values and the remaining parameters of the SSMPSP instance define an instance of MPSP. The IP formulation of this MPSP instance is solved to give the *passenger partition*, the number of passengers assigned to each class, $n_1, n_2, \ldots, n_M$.

In the assignment phase, the intervals (4.3) are used to define a policy for assigning each passenger to a class upon check-in, given the current state. Passenger $t = 1, 2, ..., N$ falls into one of the t' intervals, and this interval is matched with one of the M classes, to which passenger t is assigned. Define $\tilde{n}_i = \sum_{j=1}^{i} n_j$ as the number of passengers assigned to class $i = 1, 2, \ldots, M$ or a less secure class, with $\tilde{n}_0 = 0$. When the first passenger arrives, AT_1 becomes known and the passenger is assigned to position j_1^* (such that $J_{N,j_1^*-1} < at_1 \leq J_{N,j_1^*}$) and to class x_1 (such that $\tilde{n}_{x_1-1} < j_1^* \leq \tilde{n}_{x_1}$). The objective function value z^{SSA}, n_{x_1}, and $\tilde{n}_{x_1}, \tilde{n}_{x_1+1}, \tilde{n}_M$ are updated. This procedure is repeated for the remaining $N-1$ passengers.

SSA indirectly determines $M-1$ breakpoints at each stage in SSMPSP. Each passenger assignment is determined by the passenger's assessed threat value and the two breakpoints between which it lies. The breakpoints are dynamic, changing at each stage based on the previous passengers. The breakpoints for passenger t are $J_{t',\tilde{n}_1}, J_{t',\tilde{n}_2}, \ldots, J_{t',\tilde{n}_M}$. Passenger t is assigned to class 1 if the assessed threat value is between 0 and $J_{t',\tilde{n}_1}$ and class M if the assessed threat value is between $J_{t',\tilde{n}_M}$ and 1.

Given $J_{t',0}, J_{t',1}, \ldots, J_{t',t'}$ and $n_1, n_2, \ldots, n_M$, SSA requires $\mathrm{O}(N(\log N + M))$ time in order to find position j_t^* for passenger $t = 1, 2, \ldots, N$ and to update $\tilde{n}_1, \tilde{n}_2, \ldots, \tilde{n}_M$ based on the passenger assignments. The time complexity of computing the intervals $J_{1,0}, J_{1,1}, \ldots, J_{N,N}$ depends on the method used for computing the integrals in (4.3). However, the intervals do not need to be recomputed each time SSA is executed if $f_{\mathcal{AT}_t}(at_t)$ remains constant—the intervals can be saved for future access. Solving the IP of the corresponding MPSP instance is computationally intractable. However, optimal solutions to MPSP instances have been observed to be obtained efficiently in practice [51].

Theorem 4.1 outlines the condition under which SSA is the optimal policy. Note that the condition in Theorem 4.1 can only be checked after all of the passengers have checked in and their assessed threat values have become known. Therefore, the condition given in Theorem 4.1 only has value retrospectively.

Theorem 4.1 *If the passenger partition $n_1, n_2, \ldots, n_M$ given $E[AT_1], E[AT_2], ..., E[AT_N]$ is identical to the SSMPSP passenger partition when the true assessed threat values $at_1, at_2, ..., at_N$ are*

assumed to be known a priori, then the policy defined by SSA is the optimal policy for SSMPSP.

Proof: The proof is a direct application of Theorem 1 given by Derman et al. [18]. □

Algorithm 2 The Sequential Stochastic Assignment Heuristic

Comment: Preprocessing Phase
Compute $J_{t',0}, J_{t',1}, \ldots, J_{t',t'}$ for $t' = 1, 2, \ldots, N$ based on (4.3)
Initialize objective function value $z^{SSA} \leftarrow 0$
Solve an IP instance of MPSP with $E[AT_1], E[AT_2], \ldots, E[AT_N]$ to obtain the number of passengers to assign to each class $n_1, n_2, \ldots, n_M$
Compute $\tilde{n}_i = \sum_{j=1}^{i} n_j$, $i = 1, 2, \ldots, M$
Comment: Assignment Phase
for $t \leftarrow 1$ to N **do**
  $t' \leftarrow N - t + 1$
  at_t becomes known as passenger t arrives
  Determine position $j_t^* \in \{1, 2, \ldots, t'\}$ such that $J_{t',j_t^*-1} < at_t \leq J_{t',j_t^*}$
  Determine class $x_t \in \{1, 2, \ldots, M\}$ such that $\tilde{n}_{x_t-1} < j_t^* \leq \tilde{n}_{x_t}$
  $\tilde{n}_i \leftarrow \tilde{n}_i - 1$ for $i = x_t, x_t + 1, \ldots, M$.
  $z^{SSA} \leftarrow z^{SSA} + L_{x_t} at_t$
end for
$z^{SSA} \leftarrow z^{SSA} / (\sum_{t=1}^{N} at_t)$
return z^{SSA}, $x_1, x_2, \ldots, x_N$

4.3 Computational Results

This section reports computational results for an instance of SSMPSP that incorporates flight schedule and passenger volume data extracted from the Official Airline Guide (OAG) for the domestic flights of a single airline carrier at particular airport stations in the United States [60]. The passenger set is based on data extracted from the OAG, which includes the set of flights, the number of available seats on each flight, and the departure time of each flight. The flights departing from a single terminal at a hub airport are considered. Since some flights depart from the terminal infrequently, and including these flights would inaccurately increase the peak number of passengers, only flights with a frequency of at least three flights per week (twelve per month) are considered.

To find the peak number of passengers in the time window, passengers are assumed to arrive randomly according to a uniform distribution between 30 and 90 minutes prior to the departure time of each flight. This arrival interval is recommended by airlines for domestic flights, with 30 minutes prior to departure being the latest check-in time (e.g., see www.nwa.com, www.ual.com, www.aa.com, or www.delta.com). Moreover, each flight is assumed to have an enplanement rate

of 80% (i.e., the number of passengers divided by the number of available seats). The data set is chosen based on the largest expected number of arriving passengers in a 10 minute window, which results in $N = 916$ passengers.

The device types are chosen based on current devices and procedures used to screen passengers in commercial airports in the United States. Four devices are considered; two for screening checked baggage, and two for screening passengers and their carry-on baggage. The two devices for screening passengers and carry-on baggage are assumed to be a combination of metal detectors and X-ray machines, labeled D1, and a combination of trace portals and X-ray machines, labeled D2. The two devices for screening checked baggage are explosive detection systems (EDSs), labeled D3, and explosive trace devices, labeled D4. See Section 3.3 for an explanation for why the combination of metal detectors (trace portals) and X-ray machines is considered as one device.

Table 4.1 summarizes the false clear rates and capacities associated with the devices. The device values are estimated using information available in the public domain (see [12, 89, 52]). It is assumed that the devices for passengers operate independently of the devices for checked-baggage screening. This is reasonable since the screening for these two classes of devices is often done in different areas in airports, and the system response for checked baggage (i.e., either alarm or clear) does not depend on the system response to passenger and carry-on baggage.

Table 4.1: SSMPSP Security Device Data

Label	Device Type	False Clear	Units/hour	Units/10 minutes
D1	Metal Detector/X-ray Machine	0.20	120	20
D2	Trace Portal/X-ray Machine	0.15	90	15
D3	EDS	0.12	150	25
D4	Explosive Trace Devices	0.15	30	5

Using these devices, nine classes can be defined as follows. First, passengers can be screened in three ways: by D1, by D2, or by both D1 and D2. Likewise, checked baggage can be screened in three ways: by D3, by D4, or by both D3 and D4. The resulting nine classes enumerate all the possible ways of how passengers and baggage can be screened. The security level of the classes are the average of the true alarm rates associated with passenger screening and checked-baggage screening. Note that a different metric could be used to determine the security level of a class based on the false clear rates associated with both types of devices. Without loss of generality, assume that a passenger (checked baggage) is screened by D1 before D2 (D3 before D4) when screened by

two devices.

When passengers or checked baggage are screened by both devices, it is assumed that the system gives an alarm response if the passenger or bag triggers an alarm at either device in the system. Therefore, the system gives a clear response only if both devices give a clear response. The dependence between two screening devices is defined by (3.7). Based on the bounds in (3.8), $\varepsilon^T_{C2|C1} = \varepsilon^T_{C4|C3} = 0.1$ for the scenarios considered.

Table 4.2 contains class information, the devices used by each class, and the security level associated with each class. If device type k is (not) used by class i, then a one (zero) is listed in the Class i row and the Devices Used column k in Table 4.2. The classes are sorted such that $L_1 < L_2 < \ldots < L_9$.

Table 4.2: SSMPSP Class data

	Devices Used (d_{ik})				
Class (i)	D1 ($k=1$)	D2 ($k=2$)	D3 ($k=3$)	D4 ($k=4$)	Security Level (L_i)
1	1	0	0	1	0.825
2	1	0	1	0	0.84
3	0	1	0	1	0.85
4	0	1	1	0	0.865
5	1	0	1	1	0.885
6	1	1	0	1	0.90
7	0	1	1	1	0.91
8	1	1	1	0	0.915
9	1	1	1	1	0.96

An example with a total of 48 scenarios is considered by constructing sixteen device capacities and three assessed threat distributions. There are two levels of available capacities for each of the device types, resulting in sixteen device capacities when all combinations of available capacities are considered across the four device types. Table 4.3 summarizes the sixteen device capacity levels available in the time window.

The three assessed threat value distributions assume that most passengers can be cleared of being a threat and hence, have low assessed threat values. This is consistent with what is known in the public domain about how Secure Flight works (US GAO 2005). Recall that $\mathcal{AT}_t$ denotes the random variable that represents the assessed threat value of passenger $t = 1, 2, \ldots, N$ prior to passenger t checking in. The assessed threat values only become known upon passenger check-in. For all three types of distributions, the same distribution is used for all values of t. The first

Table 4.3: SSMPSP Device Capacity Levels

Capacity Level	D1	D2	D3	D4
1	600	375	600	375
2	600	375	600	600
3	600	375	800	375
4	600	375	800	600
5	600	600	600	375
6	600	600	600	600
7	600	600	800	375
8	600	600	800	600
9	800	375	600	375
10	800	375	600	600
11	800	375	800	375
12	800	375	800	600
13	800	600	600	375
14	800	600	600	600
15	800	600	800	375
16	800	600	800	600

assessed threat distribution is an exponential random variate with mean 1/16, where values greater than one are discarded when assessed threat values are randomly generated (labeled Type III). A randomly generated Type III assessed threat value is greater than one with a probability of approximately 10^{-7}. This distribution results in approximately 80% of the assessed threat values being less than 0.1. The second type of assessed threat distribution is a triangular distribution with probability density function

$$f_{\mathcal{AT}_t}(at_t) = 2(1 - \mathcal{AT}_t), 0 < at_t \leq 1,\ t = 1, 2, \ldots, N$$

(labeled Type IV). The third assessed threat distribution has two parts: a triangular distribution for assessed threat values less than 0.1 and a uniform distribution for assessed threat values between 0.1 and 1.0. The resulting probability density function is

$$f_{\mathcal{AT}_t}(at_t) = \begin{cases} (341 - 3400\mathcal{AT}_t)/18, & 0 \leq \mathcal{AT}_t < 0.1 \\ 1/18, & 0.1 \leq \mathcal{AT}_t \leq 1.0 \end{cases}$$

(labeled Type V). This assessed threat distribution results in 95% of the assessed threat values being less than 0.1.

The optimal policy for SSMPSP that maximizes the expected total security can be found by solving the optimality equations (4.1) and (4.2). However, finding the optimal policy in this manner is too computationally intensive for the scenarios considered. SSA provides the optimal policy under the condition in Theorem 4.1. However, this condition can only be verified retrospectively, after all N passengers have checked in.

SSA found solutions for thirty replications of each of the 48 scenarios, where new assessed threat values were randomly generated for each of the thirty replications. SSA has two phases: the preprocessing phase and the assignment phase. Since the preprocessing phase does not depend on the realized assessed threat values, the preprocessing phase did not execute for each replication. The SSA intervals (4.3) were computed once (using Matlab) for each assessed threat distribution (i.e., a total of three sets of SSA intervals were computed for the 48 scenarios). The three sets of SSA intervals were used to create 48 IPs, one corresponding to each scenario. The 48 IPs were solved in the preprocessing phase using CPLEX 9.0 on a Sun Blade 1500 with a 1.5 GHz UltraSPARC IIIi processor. Each IP was solved in fewer than 2 CPU seconds. For each of the 48 scenarios, thirty replications of the assignment phase was executed using Matlab. The Matlab computations were performed on a 2.6 GHz Pentium IV processor with 1.0 GB of RAM. The assignment phase required no more than 14 seconds for SSA for all thirty replications of each scenario.

Note that if the passenger partition for a SSMPSP solution is the same as the partition of the corresponding MPSP scenario when the assessed threat values are assumed to be known a priori, then SSA is the optimal policy. However, the subsets of passengers assigned to classes in the SSMPSP solution may be different than those of the corresponding MPSP solution since a passenger's assignment depends on the order of arrival for check-in in SSMPSP.

Table 4.4 summarizes the SSA solutions for all 48 scenarios. For each of the sixteen capacity levels, SSA found the same partition across all three assessed threat distributions. Moreover, the partitions for all thirty replications of each of the 48 scenarios are the same as the corresponding MPSP partitions when the assessed threat values are assumed to be known a priori. Therefore, SSA is the optimal policy for all 48 scenarios considered. Table 4.5 reports the average solution values and the standard deviation of the objective function values across all thirty replications of all 48 scenarios. The small variation in the SSA solution values suggests that SSA is robust in assigning passengers to classes. These SSMPSP objective function values are within one-percent of the corresponding MPSP objective function values when the assessed threat values are assumed

to be known a priori.

Table 4.4: SSA Solutions for Type III, IV, V Assessed Threat Distributions

	Passenger Partition								
Capacity Level	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
1	316	225	0	316	0	0	0	0	59
2	316	225	0	91	0	0	225	0	59
3	116	425	0	116	0	0	200	0	59
4	116	316	0	0	109	0	316	0	59
5	316	0	0	316	0	0	0	225	59
6	316	0	0	316	0	0	0	0	284
7	116	200	0	316	0	0	0	25	259
8	116	200	0	116	0	0	200	0	284
9	316	225	0	116	0	0	0	200	59
10	316	225	0	91	0	0	25	0	259
11	116	425	0	116	0	0	0	0	259
12	116	316	0	0	109	0	116	0	259
13	316	0	0	116	0	0	0	425	59
14	316	0	0	116	0	0	0	200	284
15	116	200	0	116	0	0	0	225	259
16	116	200	0	116	0	0	0	0	484

One goal of multilevel passenger screening systems is to direct more security resources toward passengers perceived as higher-risk [58]. However, the objective function value captures the expected total security, not the accuracy of SSA in assigning higher-risk passengers to more secure classes. Since there are few higher-risk passengers, their effect on the objective function value may be diluted by the more numerous lower-risk passengers. SSA indirectly determines $M-1$ breakpoints during each stage $t = 1, 2, \ldots, N$, with passengers assigned to classes based on which breakpoints their assessed threat values lie between. The breakpoints change after every passenger arrival.

The scenario with capacity level 6 and the Type III assessed threat value is used to illustrate how SSA assigns passengers to classes. SSA assigns 284 passengers to class 9 (see Table 4.4). Ideally, the 284 passengers with the highest assessed threat values should be assigned to class 9. Note that there are only two unique breakpoints between the classes, since only three of the classes are used in the solutions.

Figure 4.1 shows the average breakpoints over the thirty replications to the Type III scenario with capacity level 6. The breakpoints are relatively constant for most of the time period. Both

Table 4.5: Summary of SSA Solution Values

	Type III		Type IV		Type V	
Capacity Level	Average Value	Standard Deviation	Average Value	Standard Deviation	Average Value	Standard Deviation
1	0.871	0.0006	0.905	0.0042	0.906	0.0036
2	0.890	0.0007	0.915	0.0027	0.917	0.0026
3	0.890	0.0007	0.915	0.0033	0.916	0.0025
4	0.902	0.0006	0.923	0.0031	0.925	0.0020
5	0.897	0.0006	0.921	0.0023	0.922	0.0022
6	0.917	0.0009	0.931	0.0016	0.934	0.0020
7	0.916	0.0007	0.931	0.0015	0.933	0.0017
8	0.928	0.0007	0.940	0.0015	0.941	0.0014
9	0.890	0.0008	0.916	0.0033	0.918	0.0022
10	0.910	0.0010	0.927	0.0024	0.930	0.0022
11	0.909	0.0011	0.927	0.0022	0.930	0.0019
12	0.921	0.0006	0.935	0.0017	0.937	0.0014
13	0.909	0.0006	0.928	0.0020	0.929	0.0019
14	0.928	0.0008	0.940	0.0017	0.941	0.0013
15	0.928	0.0007	0.939	0.0014	0.940	0.0018
16	0.940	0.0007	0.947	0.0011	0.948	0.0010

breakpoints appear to decrease at the end of the time period, which indicates that passengers who check in at the end of the time period are more likely to be assigned to more secure classes. In all SSA replications, it was observed that the last passenger to check in is always assigned to class 9, regardless of the passenger's assessed threat value. The SSA policy appears to save available space in class 9 for high-risk passengers that may check in at the end of the time period. Therefore, SSA provides a policy that is difficult for high-risk passengers to game—extremely high-risk passengers are always assigned to class 9 in the scenarios considered. The average breakpoints look similar for the Type IV and V distributions, with the breakpoints rescaled with respect to the assessed threat values.

To quantify the worst-case performance of SSA, two extreme cases are considered: passengers who check in with increasing assessed threat values and passengers who check in with decreasing assessed threat values. These two cases are compared to the original scenarios when passengers are assumed to arrive at random. The case with increasing (decreasing) assessed threat values represents the scenario when lower-risk passengers arrive first (last), and higher-risk passengers arrive last (first). The case with increasing assessed threat values provides insight on whether a large number of lower-risk passengers arriving early in the time period use scarce screening

resources associated with class 9, which are ideally reserved for the higher-risk passengers who arrive later. The case with decreasing assessed threat values provides insight on whether a large number of higher-risk passengers who arrive early in the time period are assigned to less secure classes in order to keep resources associated with class 9 available for higher-risk passengers that are expected to arrive later in the time period.

Figure 4.2 shows the average breakpoints across thirty replications for the cases with increasing and decreasing assessed threat values. The breakpoints increase near the end of the time period for case with increasing assessed threat values. This indicates that some high-risk passengers would be screened with the least secure class, and hence, it is possible for terrorists to game the system by altering when passengers check-in. However, this would be difficult to achieve in practice, particulary at a hub airport with a large number of passengers. SSA is more robust in assigning high-risk passengers to class 9 for the case with decreasing assessed threat values. This result is counterintuitive since the case with random passenger arrivals suggests that SSA is most effective in screening high-risk passengers who check in at the end of the time period, not at the beginning, while in the case with decreasing assessed threat values, the high-risk passengers who check in at the beginning of the time period are consistently assigned to class 9.

4.4 Conclusions

Passenger screening is a critical component of any aviation security system operation. This chapter introduces SSMPSP, which models stochastic passenger and baggage screening systems. SSMPSP, an extension of MPSP that considers passengers arriving over time, is formulated as a MDP, where the optimal policy can be computed by dynamic programming. The SSA heuristic is presented to provide approximate solutions to SSMPSP in real-time. A condition is provided under which SSA yields the optimal solution. SSA obtains solutions to SSMPSP in real-time, and this suggests that SSA could be part of a tool used by the TSA in assigning passengers to classes in real-time.

Data extracted from the OAG was used to construct the passenger set for the SSMPSP example with three assessed threat distributions and sixteen capacity levels (and hence, a total of 48 scenarios). It was verified retrospectively that SSA provided the optimal policy for all 48 scenarios. Analysis suggests that SSA is almost certain to assign extremely high-risk passengers to the most secure class, regardless of when these passengers check in. An analysis of two extreme cases, passengers arriving with increasing and decreasing assessed threat values, suggests that there may

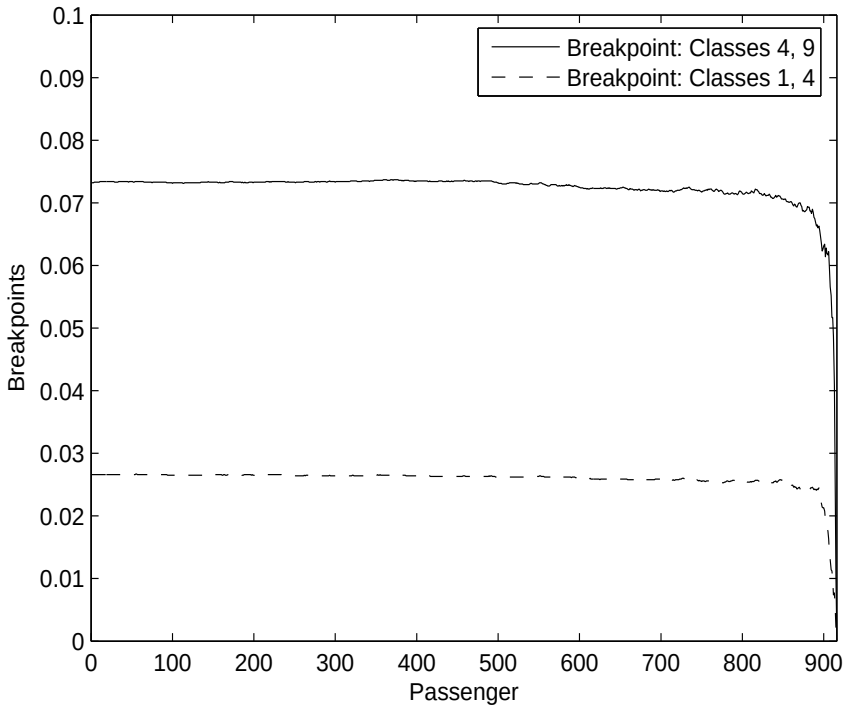

Figure 4.1: Average Type III Breakpoints

be scenarios in which SSA can be gamed if the order in which passengers check in is skewed.

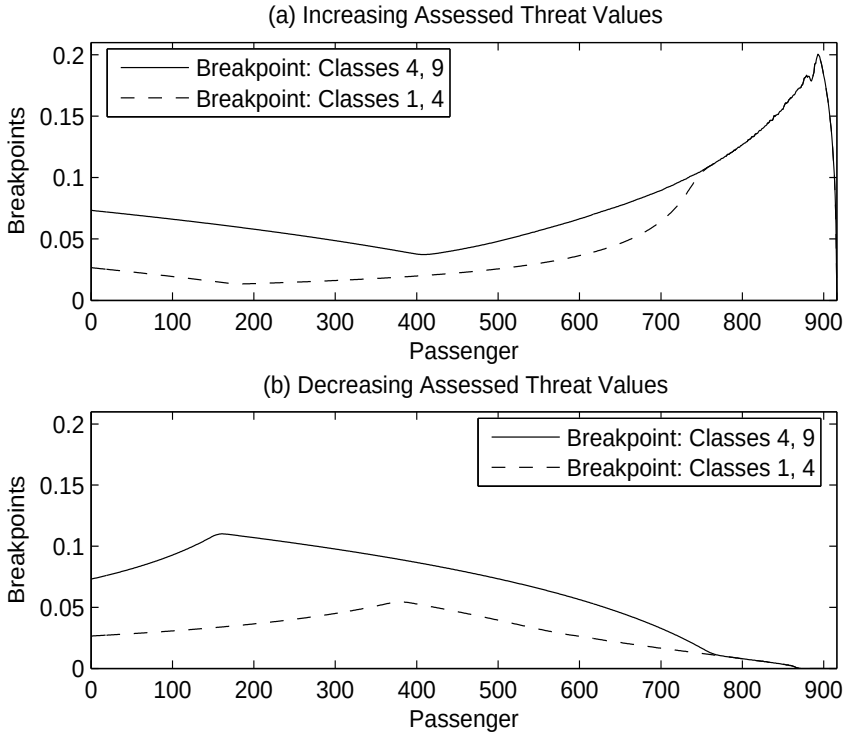

Figure 4.2: Average Type III Breakpoints when Assessed Threat Values are Increasing and Decreasing

Chapter 5

A Cost Benefit Analysis for Selective Checked Baggage Screening Systems

An effective selective screening strategy for aviation security depends on two criteria: the ability of the prescreening system to correctly identify threat passengers as selectees and the effectiveness of security screening for selectees and nonselectees. This chapter considers both criteria in association using a cost-benefit analysis of selective screening strategies for passenger baggage. The key contribution of the analysis in this chapter is that the prescreening system must be accurate in assessing passenger risk in order to reduce the number of successful attacks, particularly if the proportion of passengers classified as selectees is small. Moreover, the analysis indicates that when a large proportion of passengers are classified as selectees, it may not be cost-effective to use selective screening.

This chapter is organized as follows. Section 5.1 provides a description of the data needed to describe the risk and cost models. Section 5.2 presents the risk and cost models that quantify the effectiveness and the costs of a selective screening system. Section 5.3 analyzes these risk and cost models, and reports results for a number of different scenarios. Section 5.4 provides concluding comments.

5.1 Data Needs

Data is needed to support the risk and cost models used in the analysis in Sections 5.2 and 5.3. These data are represented as parameters that can be classified into three groups: (1) probability parameters, (2) cost parameters, and (3) other parameters.

(1) Probability Parameters

$P_{\mathrm{T}} =$ probability that a checked bag contains a threat

$P_{\mathrm{FA}} = P_{\mathrm{A|NT}} =$ probability that the baggage screening device falsely indicates a threat (*false alarm*)

$P_{\mathrm{TC}} = P_{\mathrm{NA|NT}} = 1 - P_{\mathrm{FA}} =$ probability of a *true clear*

$P_{\mathrm{TA}} = P_{\mathrm{A|T}} =$ probability that the baggage screening device correctly indicates a threat (*true alarm*)

$P_{\mathrm{FC}} = P_{\mathrm{NA|T}} = 1 - P_{\mathrm{TA}} =$ probability of a *false clear*.

The probability that a checked bag contains a threat is a random variable that is assessed by personnel within the TSA based on the perceived threat level. This value is considered highly sensitive and may change based on changes in national or international situations, intelligence information, or the risk level of the Homeland Security Advisory System [76]. The final four probability parameters are random variables obtained based on the testing and evaluation of a baggage screening security device; such testing is required before a device can gain TSA certification.

(2) Cost Parameters

$C_{\mathrm{FA}} =$ cost of a *false alarm* = cost of falsely indicating a threat

$C_{\mathrm{TC}} =$ cost of a *true clear* = cost of correctly indicating a non-threat

$C_{\mathrm{TA}} =$ cost of a *true alarm* = cost of correctly indicating a threat

$C_{\mathrm{FC}} =$ cost of a *false clear* = cost of not detecting a threat

$C_F =$ purchase price and installation cost (fixed) of a baggage screening security device

$C_O =$ annual maintenance costs (operational) for a baggage screening security device, including annual lease expenses. This cost is independent of the volume of checked bags screened by the device.

$C_I =$ cost of operating a baggage screening security device per checked bag inspected.

The first four cost parameters are random variables based on information collected and analyzed by the TSA. The next two cost parameters are deterministic and are available from the baggage screening security device manufacturer. Note that in most airports, baggage screening security devices are not in use 24 hours per day. Therefore, ample non-operational time is available for maintenance such that breakdowns rarely occur. This is reflected in the high maintenance costs for such devices. The last cost parameter is also deterministic, obtained from the TSA based on

salaries paid to the federal employees hired to operate the baggage screening security devices.

(3) Other Parameters

$N =$ total number of passengers at a given station in a year

$L_{eff} =$ effective lifetime (in years) of a baggage screening device

$S_{CAP} =$ the number of checked bags the baggage screening security device can process per year (screening capacity).

The first parameter is a deterministic value that represents the number of passengers that pass through a given *station* in a year. A station is a set of airport facilities that share security resources. For example, a station could be an entire small airport or one airline's terminal within a larger airport. The effective lifetime of a baggage screening device is the minimum of three values: the useful life of the device before technical obsolescence, the number of years until the device wears out, and the number of years until the maximum allowable number of bags has been processed. The second and third parameters are deterministic values based on the testing and evaluation of a baggage screening device. The effective lifetime is deterministic since the TSA is likely to replace all baggage screening devices after a given amount of time rather than wait for them to wear out.

This chapter considers aviation security strategies that use two types of baggage screening security devices: one type of baggage screening device to screen selectee baggage and another type of baggage screening device to screen nonselectee baggage. To differentiate between the parameters for both selectees and nonselectees, the symbol S is used in conjunction with the parameters for selectees, and NS is used in conjunction with the parameters for nonselectees. Furthermore, to differentiate between the parameters for both types of screening devices, the symbol DS is used in conjunction with the parameters for the baggage screening devices for selectees, and DNS is used in conjunction with the parameters for the baggage screening devices for nonselectees. For example, the probability that a threat passenger is classified as a selectee is denoted by $P_{\mathrm{S|T}}$, and the expected inspection cost for nonselectee baggage is denoted by $C_{\mathrm{I,DNS}}$.

5.2 Risk and Cost Models

This section describes the risk and cost models used in the analysis in Section 5.3. To apply these models, first assume that passengers are independent and that every passenger has exactly one checked bag. Moreover, assume that the capacity of a single baggage screening security device is based on it being in use six hours a day (i.e., peak hours of operation), 360 days per year.

A prescreening system classifies each passenger as either a selectee or a nonselectee. Each flight is composed of a mixture of selectees and nonselectees, which varies according to the flight and passenger characteristics. For example, a commuter flight may only have nonselectees, while an international flight may have a comparable number of selectees and nonselectees. However, given a large number of passengers across many flights, a certain proportion of passengers P_S is expected to be classified as selectees based on how the prescreening system is designed and operated.

The *risk model* captures the ability of the prescreening system to correctly identify threat passengers as selectees. Ideally, all threat passengers are classified as selectees. The probability that a threat passenger is classified as a selectee is given by $P_{\mathrm{S|T}}$. Similarly, the probability that a non-threat passenger is classified as a selectee ($P_{\mathrm{S|NT}}$) is given by

$$P_{\mathrm{S|NT}} = \frac{P_{\mathrm{S,NT}}}{P_{\mathrm{NT}}} = \frac{P_S - P_{\mathrm{S|T}} P_{\mathrm{T}}}{1 - P_{\mathrm{T}}},$$

and the probability that a non-threat passenger is a nonselectee ($P_{\mathrm{NS|NT}}$) is given by

$$P_{\mathrm{NS|NT}} = 1 - P_{\mathrm{S|NT}}.$$

Using Bayes' Rule, the probability that a selectee is a threat ($P_{\mathrm{T|S}}$) is given by

$$P_{\mathrm{T|S}} = P_{\mathrm{S|T}} P_{\mathrm{T}} / P_S,$$

and the probability that a nonselectee is a threat ($P_{\mathrm{T|NS}}$) is given by

$$P_{\mathrm{T|NS}} = \frac{P_{\mathrm{NS|T}} P_{\mathrm{T}}}{1 - P_S},$$

where $P_{\mathrm{NT|S}} = 1 - P_{\mathrm{T|S}}$ and $P_{\mathrm{NT|NS}} = 1 - P_{\mathrm{T|NS}}$.

The ratio of the proportion of threats classified as selectees to the proportion of threats classified as nonselectees is called the *prescreening multiplier*,

$$\beta = \frac{P_{\mathrm{T|S}}}{P_{\mathrm{T|NS}}}.$$

It is reasonable to assume that $\beta \geq 1$, since the prescreening system is designed to identify passengers (and their checked baggage) who are unlikely to be threats to the system [53]. Using the

equations for $P_{\mathrm{T|S}}$ and $P_{\mathrm{T|NS}}$ yields the expression for the risk model,

$$P_{\mathrm{S|T}} = \frac{\beta P_S}{1 - P_S + \beta P_S}. \tag{5.1}$$

Note that $P_{\mathrm{S|T}} = 0$ when $P_S = 0$ (i.e., no passengers are designated as selectees) and $P_{\mathrm{S|T}} = 1$ when $P_S = 1$. Moreover, when $\beta = 1$, then $P_{\mathrm{S|T}} = P_S$, and therefore, $\beta = 1$ corresponds to the scenario when all passengers are randomly classified as selectees or nonselectees.

The *cost model* is a random variable that captures the annual cost of a given baggage screening strategy, including both direct and indirect costs. Direct costs include the annual purchasing, maintaining, and operating costs of the baggage screening security devices, as well as the direct costs associated with processing the volume of true clears, addressing true alarms (including the possible closing of an airport terminal), and resolving false alarms (including the possible need to call in law enforcement officers and bomb squads). Indirect costs are those associated with false clears. Since such incidents are typically rare events, and hence the time between them is long, then this cost may only occur once every 10 to 20 years. The cost model takes this into account by evenly spreading these costs over such a time period.

The expected value of the annual cost is computed using the expected value for the parameters in the cost model. Note that since the cost parameters and the probability parameters in (5.2) that are multiplied are all independent, then the expected cost can be computed using the expectation of the parameters.

The cost model assumes that there are two sets of baggage screening devices and procedures, one for screening selectees and the other for screening nonselectees. The expected numbers of baggage screening devices for selectees and nonselectees, D_{DS} and D_{DNS}, depend on the expected number of passengers designated as selectees and nonselectees, respectively. These values are given by

$$D_{\mathrm{DS}} = \lceil N P_S / S_{\mathrm{CAP,DS}} \rceil$$

and

$$D_{\mathrm{DNS}} = \lceil N(1 - P_S) / S_{\mathrm{CAP,DNS}} \rceil,$$

respectively.

Using this information, the cost model is as follows:

$$\text{Cost} = D_{DNS}(C_{F,DNS}/L_{eff,DNS} + C_{0,DNS}) + D_{DS}(C_{F,DS}/L_{eff,DS} + C_{0,DS}) +$$

$$N(1 - P_S)C_{I,DNS} + N\,P_S C_{I,DS} + (N_{\text{NT,NS}}P_{\text{FA|DNS}} + N_{\text{NT,S}}P_{\text{FA|DS}})C_{FA} + \tag{5.2}$$

$$(N_{\text{T,NS}}P_{\text{FC|DNS}} + N_{\text{T,S}}P_{\text{FC|DS}})C_{FC} + (N_{\text{T,NS}}P_{\text{TA|DNS}} + N_{\text{T,S}}P_{\text{TA|DS}})C_{TA} +$$

$$(N_{\text{NT,NS}}P_{\text{TC|DNS}} + N_{\text{NT,S}}P_{\text{TC|DS}})C_{TC}$$

In the cost model (5.2),

- the first and second components represent the annual direct costs of purchasing and maintaining the devices for screening nonselectees and selectees, respectively,
- the third and fourth components represent the annual baggage inspection direct cost for the devices for nonselectees and selectees, respectively,
- the fifth component represents the annual direct cost of false alarms,
- the sixth component represents the annual indirect cost of false clears,
- the seventh component represents the annual direct cost of true alarms, and
- the eighth component represents the annual direct cost of true clears.

The expected annual cost is computed using the expected values for the parameters that are random variables are given in Table 5.1. Note that not all of the values in Table 5.1 are real-world actual data—since some of these values are security-sensitive information, they cannot be disseminated in the public domain. However, the values in Table 5.1 are representative of the real-world values.

When the expected annual cost is computed, then the expected number of non-threat, nonselectees is given by

$$N_{\text{NT,NS}} = N(1 - P_{\text{T}})P_{\text{NS|NT}},$$

the expected number of non-threat, selectees is given by

$$N_{\text{NT,S}} = N(1 - P_{\text{T}})P_{\text{S|NT}},$$

Table 5.1: Parameter Expected Values

Parameters	Expected Value
$P_{\text{FA}\|\text{DNS}}$, $P_{\text{FA}\|\text{DS}}$	0.30
$P_{\text{FC}\|\text{DNS}}$	0.05
P_{T}	5.005×10^{-9}
C_{FC}	\$1.4B
C_{TA}	\$1M
C_{TC}	\$0
C_{FA}	\$9
$C_{\text{F,NS}}$	\$1M
$C_{\text{O,NS}}$	\$125K
$C_{\text{I,NS}}$	\$1

the expected number of threat, nonselectees is given by

$$N_{\text{T,NS}} = N\, P_{\text{T}} P_{\text{NS}|\text{T}},$$

and the expected number of threat, selectees is given by

$$N_{\text{T,S}} = N\, P_{\text{T}} P_{\text{S}|\text{T}}.$$

The cost of a false clear, C_{FC}, is both large and difficult to estimate. Therefore, a second cost model is formulated that only considers direct costs. The resulting *direct cost model* represents the annual direct cost of operating baggage screening security devices (it uses all but the sixth term in (5.2)) is as follows:

$$\begin{aligned}\text{Cost}_D = {} & D_{DNS}(C_{F,DNS}/L_{eff,DNS} + C_{0,DNS}) + D_{DS}(C_{F,DS}/L_{eff,DS} + C_{0,DS}) + \\ & N(1-P_S)C_{I,DNS} + N\,P_S C_{I,DS} + (N_{\text{NT,NS}}P_{\text{FA}|\text{DNS}} + N_{\text{NT,S}}P_{\text{FA}|\text{DS}})C_{FA} + \\ & (N_{\text{T,NS}}P_{\text{TA}|\text{DNS}} + N_{\text{T,S}}P_{\text{TA}|\text{DS}})C_{TA} + (N_{\text{NT,NS}}P_{\text{TC}|\text{DNS}} + N_{\text{NT,S}}P_{\text{TC}|\text{DS}})C_{TC}\end{aligned} \tag{5.3}$$

The expected annual direct cost is computed using the expected value for the parameters in the cost model.

A third measure that can be used to quantify the effectiveness of a security screening strategy is the number of successful attacks. Define the number of successful attacks as

$$\text{NSA} = N_{\text{T,NS}}P_{\text{FC}|\text{DNS}} + N_{\text{T,S}}P_{\text{FC}|\text{DS}}.$$

The number of successful attacks can be normalized by the number of passengers to compare different scenarios. The *number of successful attacks per billion passengers* is a random variable given by

$$\mathrm{NSA}_B = (N_{\mathrm{T,NS}} P_{\mathrm{FC|DNS}} + N_{\mathrm{T,S}} P_{\mathrm{FC|DS}}) \frac{10^9}{N}. \tag{5.4}$$

A fourth measure that can be used to quantify the effectiveness of a security screening strategy is the *expected direct cost to prevent an attack*, which compares the selective screening strategies (SS) to a base case (BC). The base case, defined in Section 4, screens all checked baggage with nonselectee checked baggage screening devices (EDSs). The expected direct cost to prevent an attack (EPA) is defined as the difference in the expected direct costs divided by the difference in the expected number of successful attacks between the selective screening and base case strategies,

$$\mathrm{EPA} = \frac{\mathrm{Cost}_D^{SS} - \mathrm{Cost}_D^{BC}}{\mathrm{NSA}^{BC} - \mathrm{NSA}^{SS}}. \tag{5.5}$$

The expected direct cost to prevent an attack quantifies the additional expected direct cost to prevent a single attack, compared to the base case. Note that there may be dependencies between the direct cost and number of successful attacks since $P_{\mathrm{TA|DS}} = 1 - P_{\mathrm{FC|DS}}$ and $P_{\mathrm{TA|DNS}} = 1 - P_{\mathrm{FC|DNS}}$. This measure provides one way to quantify the *cost-effectiveness* of a baggage screening strategy.

5.3 Cost Model Analysis

The risk model (5.1), the direct cost model (5.3), the number of successful attacks per billion passengers (5.4), and the expected direct cost to prevent an attack (5.5), provide measures to quantify the performance of selective aviation security strategies using prescreening under different operating conditions and assumptions. Using the risk model, the value of the prescreening multiplier β determines the probability that a threat is classified as a selectee for a given proportion of passengers classified as selectees P_S. Before September 11, 2001, P_S was reported to be approximately 0.05 [2]. In this case, if $\beta = 10$, then $P_{S|T} = 0.344$, which would correspond to six of the nineteen terrorists on September 11, 2001 classified as selectees. The actual number of the nineteen terrorists classified as selectees on September 11, 2001 has been reported to be at least six and as many as eleven [3, 56, 71]. Therefore, a realistic lower-bound for β for an improved prescreening system, such as Secure Flight, is ten. Moreover, if $\beta = 100$, then approximately sixteen of the nineteen

terrorists on September 11, 2001 would have been classified as selectees.

The analysis considers two different sets of checked baggage screening devices and procedures, one for screening nonselectee baggage and another for screening selectee baggage. At present, all checked baggage are screened by EDSs or alternative TSA certified technologies [54]. Given that the TSA is currently pursuing the development of baggage screening devices that improve upon EDSs [55], assume that nonselectee baggage is screened by EDSs, and that selectee baggage is screened by a more effective baggage screening device.

A more effective baggage screening device reduces the probability of a false clear as compared to EDSs. Suppose that the baggage screening devices for selectees decrease the false clear rate as compared to EDSs by a factor of α, with $0 < \alpha \leq 1$ (i.e, $P_{\mathrm{FC|DS}} = \alpha P_{\mathrm{FC|DNS}}$). It is assumed that as a baggage screening device becomes more effective (i.e., α decreases), the screening costs increase. The analysis considers three relationships between screening costs and the probability of a false clear. For all three relationships, the baggage screening devices for selectees are assumed to have the same effective lifetime and false alarm rate as EDSs (i.e., $L_{eff,DS} = L_{eff,DNS}$ and $P_{\mathrm{FA|DS}} = P_{\mathrm{FA|DNS}}$). Therefore, both types of baggage screening devices have the same true clear rates (i.e, $P_{\mathrm{TC|DS}} = P_{\mathrm{TC|DNS}}$). For the first relationship (labeled *Relationship 1*), the purchase and installation cost, the maintenance cost, and the inspection cost associated with the baggage screening devices for selectees increase by a factor of $1/\alpha$ as compared to EDSs (i.e, $C_{\mathrm{F,DS}} = C_{\mathrm{F,DNS}}/\alpha$, $C_{\mathrm{O,DS}} = C_{\mathrm{O,DNS}}/\alpha$, and $C_{\mathrm{I,DS}} = C_{\mathrm{I,DNS}}/\alpha$). For the second relationship (labeled *Relationship 2*), the purchase and installation cost, the maintenance cost, and the inspection cost associated with the baggage screening devices for selectees increase by a factor of $1/\sqrt{\alpha}$ as compared to EDSs (i.e, $C_{\mathrm{F,DS}} = C_{\mathrm{F,DNS}}/\sqrt{\alpha}$, $C_{\mathrm{O,DS}} = C_{\mathrm{O,DNS}}/\sqrt{\alpha}$, and $C_{\mathrm{I,DS}} = C_{\mathrm{I,DNS}}/\sqrt{\alpha}$). For the third relationship (labeled *Relationship 3*), the purchase and installation cost, the maintenance cost, and the inspection cost associated with the baggage screening devices for selectees increase by a factor of $1/\alpha^2$ as compared to EDSs (i.e, $C_{\mathrm{F,DS}} = C_{\mathrm{F,DNS}}/\alpha^2$, $C_{\mathrm{O,DS}} = C_{\mathrm{O,DNS}}/\alpha^2$, and $C_{\mathrm{I,DS}} = C_{\mathrm{I,DNS}}/\alpha^2$).

The relationships between α and the costs, the lifetime, and the probabilities represent three possible ways to compare and contrast the two types of baggage screening devices. Moreover, such relationships are reasonable since baggage screening devices that are more accurate with respect to recognizing threats (i.e., devices that reduce the probability of a false clear) are also likely to be more expensive to purchase, operate, and maintain. Moreover, an improvement in the false clear rate in such devices is usually at the expense of the false alarm rate. Therefore, assuming that

the false alarm rate is the same for both types of devices (i.e., rather than assuming that the false alarm rate of the baggage screening device for selectees decreases as compared to that of EDSs) provides a conservative lower bound on the false alarm rate associated with the security devices for selectees.

The risk model (5.1) provides a measure for comparing the relative effectiveness of a prescreening system, given by values of $\beta > 1$, as compared to the base case and the random screening cases. The *base case* is the current strategy employed by the TSA of 100% baggage screening of checked baggage by EDSs. This scenario corresponds to $\alpha = 1$ with the baggage screening devices for selectees being the same as EDSs for all values of β. Moreover, the base case assumes that no selective screening information is available (i.e., $\beta = 1$). The *random screening* cases screen a portion of checked baggage using the more effective baggage screening device. These scenarios correspond to $\beta = 1$ with various values of $0 < \alpha < 1$, with EDSs screening all remaining checked baggage. Note that a prescreening system is not required for the random screening strategy.

Since β is a function of P_S, it is difficult to compare scenarios with a given β across different values of P_S. As P_S increases for a fixed value of $\beta > 1$, the ratio of the number of threat passengers classified as selectees to the number of threat passengers classified as nonselectees is constant. However, $P_{S|T}$ increases as a result of more passengers being classified as selectees, not as a result of an improvement in intelligence. To avoid this problem, scenarios with a fixed value of P_S are compared across different values of β. In this case, $P_{S|T}$ increases as β increases as a result of improvements in intelligence information.

The measures (5.3), (5.4), and (5.5), are analyzed across several scenarios, with various values of α, β, and P_S using Relationships 1, 2, 3. The results are compared to those of the base case and the random screening cases. The three values of the prescreening multiplier considered are $\beta = 1,\ 10,\ 100$ (where $\beta = 1$ corresponds to the random screening case), and the three values of P_S considered are 0.05, 0.10, 0.20. The scenarios use the expected values from Table 5.1 and consider the passengers originating at a single station at an airport over the course of one year with $N = 10$ million passengers. The effective lifetime and the screening capacity of EDSs are deterministic, having values of 10 years 125 bags per hour, respectively.

Table 5.2: Expected Direct Cost per Passenger

		Cost_D per Passenger ($)		
α	Relationship	$P_S = 0.05$	$P_S = 0.10$	$P_S = 0.20$
0.33	1	4.75	4.95	5.33
	2	4.63	4.70	4.84
	3	5.34	6.11	7.67
0.67	1	4.61	4.65	4.75
	2	4.58	4.60	4.64
	3	4.68	4.79	5.03

5.3.1 Analysis of the Direct Cost Model

The expected direct cost per passenger is analyzed for $\alpha = 0.33,\ 0.67,\ 1.0$, $P_S = 0.05$, 0.10, 0.20, and for Relationships 1, 2, 3. The *expected direct cost per passenger* is computed as the expected direct cost (5.3) divided by the number of passengers. Each value in Table 5.2 corresponding to a different value of α reports the expected direct cost per passenger for $\beta = 1,\ 10,\ 100$, since the differences in expected direct cost for the different values of β are insignificant. The expected direct cost of the base case is \$4.56 per passenger for $\alpha = 1$ and all values of β and P_S. Note that when $P_S = 0.05$, there is little difference in the expected directed costs per passenger between the scenarios (i.e., a scenario corresponds to fixed values of α, β, and P_S). The increase in expected direct cost per passenger becomes more pronounced as P_S increases.

The relationship between the expected direct cost per passenger and P_S is approximately linear. The expected direct cost per passenger (y) can be estimated for each value of α as

$$y = a + b(\alpha)P_S,$$

where a is the y-intercept and $b(\alpha)$ is the slope that depends on α. Note that $a = \$4.56$ for all values of α since the y-intercept corresponds to screening all checked baggage with EDSs, and $b(1) = 0$. Table 5.3 shows the values of $b(\alpha)$ for several scenarios. The slope represents the marginal increase in expected direct cost per passenger as P_S increases. For example, if P_S increases by 0.01 for Relationship 3 with $\alpha = 0.33$, then the expected direct cost per passenger increases by \$0.152.

5.3.2 Analysis of the Expected Number of Successful Attacks

Table 5.4 and Figure 5.1 report the expected number of successful attacks per billion passengers for different combinations of α, β, and P_S. In Figure 5.1, P_S is labeled "P[S]", α is labeled "alpha," and β is labeled "beta." Note that for each combination of α and β, Relationships 1, 2, 3 yield the

Table 5.3: Slope Values for the Expected Direct Cost per Passenger

α	Relationship	$b(\alpha)$
0.33	1	3.8
	2	1.4
	3	15.2
0.67	1	0.91
	2	0.41
	3	2.3

Table 5.4: Expected Number of Successful Attacks per Billion Passengers

		NSA_B		
α	β	$P_S = 0.05$	$P_S = 0.10$	$P_S = 0.20$
0.33	1	0.24	0.23	0.22
	10	0.19	0.16	0.13
	100	0.11	0.10	0.09
0.67	1	0.25	0.24	0.23
	10	0.22	0.21	0.19
	100	0.18	0.17	0.17

same expected number of successful attacks since the probability of a false clear is the same. When $\alpha = 1$, the expected number of successful attacks per billion passengers is 0.25 for all values of P_S (the base case). Note that for all scenarios using different baggage screening devices for selectees (i.e., $\alpha < 1$), the expected number of successful attacks decreases, as compared to the base case.

Table 5.4 and Figure 5.1 suggest that the prescreening system must be accurate (i.e., $\beta > 1$) in order to significantly reduce the number of successful attacks. Moreover, given P_S, improving the effectiveness of baggage screening devices for selectees is not sufficient to reduce the expected number of successful attacks; the prescreening system must also be accurate. When $P_S = 0.05$ and $\alpha = 0.33$, the expected number of successful attacks per billion passengers is reduced by 3.2% for $\beta = 1$, 23.2% for $\beta = 10$, and 56.4% when $\beta = 100$. When $P_S = 0.05$, the scenario with $\alpha = 0.67$ and $\beta = 100$ has fewer expected successful attacks than the scenarios with $\alpha = 0.33$ and $\beta \leq 10$. This suggests that the accuracy of the prescreening system may be more critical for detecting attacks for small values of P_S than the effectiveness of baggage screening devices for selectees. However, the baggage screening devices for selectees asymptotically limit the expected number of successful attacks as $\beta \to +\infty$ (i.e., the expected number of successful attacks is at least 0.168 per billion passengers for $\alpha = 0.67$ and at least 0.0823 for $\alpha = 0.33$).

Figure 5.2 reports the expected number of successful attacks per billion passengers versus the expected direct cost per passenger for $P_S = 0.05$, 0.10, for the Relationship 1 scenarios, where α varies from 0.05 to 1.0 in increments of 0.05. In Figure 5.2, P_S is labeled "P[S]" and β is labeled

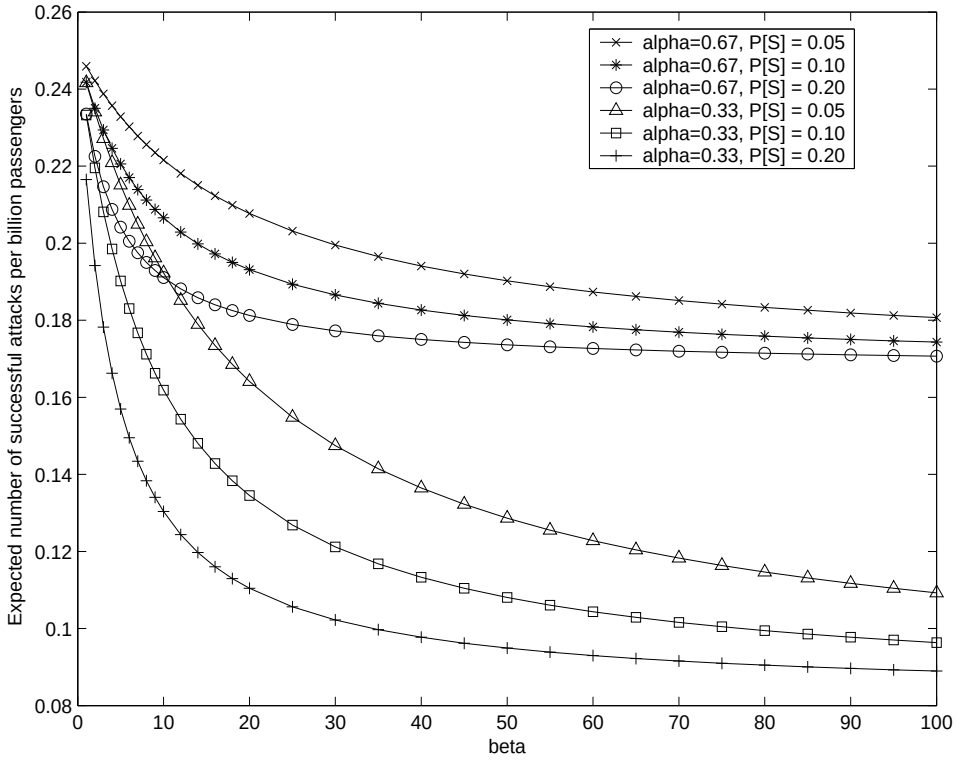

Figure 5.1: Expected Number of Successful Attacks

"beta." The cost values are determined by P_S, β, and α, and hence, increasing the expected direct cost per passenger beyond the range given can be achieved as α approaches zero. For the base case when $\alpha = 1$, the expected direct cost is \$4.56 per passenger with 0.25 expected successful attacks per billion passengers. The random screening strategies (i.e., $\beta = 1$) do not significantly reduce the expected number of successful attacks as the expected direct cost per passenger increases. For $P_S = 0.05$ and $\beta = 1$, the expected number of successful attacks per billion passengers decreases to 0.238 for $\alpha = 0.05$, while the expected direct cost per passenger increases to \$6.37. For $P_S = 0.10$ and $\beta = 1$, the expected number of successful attacks per billion passengers decreases to 0.226 for $\alpha = 0.05$, while the expected direct cost per passenger increases to \$8.17.

When the prescreening system is accurate (i.e., $\beta > 1$), the expected number of successful attacks decreases significantly for small increases in the expected direct cost per passenger. When $P_S = 0.05$ and the expected direct cost per passenger increases to \$4.65, corresponding to $\alpha = 0.5$, the expected number of successful attacks per billion passengers decreases to 0.207 for $\beta = 10$ and 0.145 for $\beta = 100$. When $P_S = 0.05$ and the expected direct cost per passenger increases to \$4.84, corresponding to $\alpha = 0.25$, the expected number of successful attacks per billion passengers

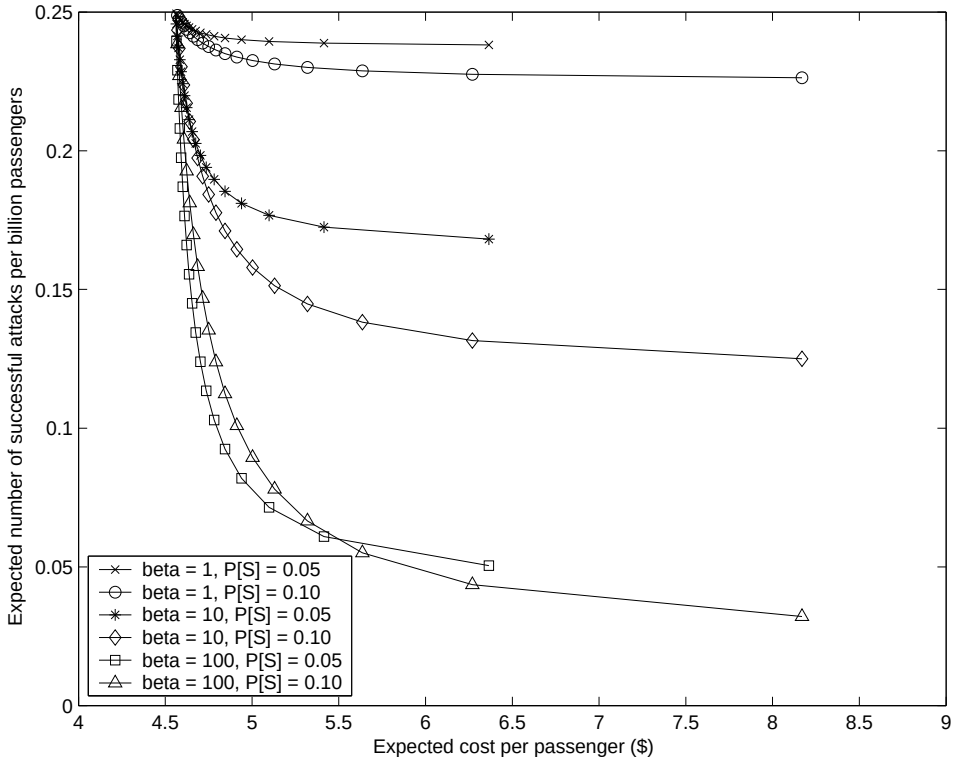

Figure 5.2: Expected Number of Successful Attacks versus Expected Direct Cost

decreases to 0.185 for $\beta = 10$ and 0.092 for $\beta = 100$. Note that similar observations can be made for the $P_S = 0.20$ scenarios, as well as for the Relationship 2 and 3 scenarios, where the expected number of successful attacks are rescaled with respect to the expected direct cost.

5.3.3 Analysis of the Expected Direct Cost to Prevent an Attack

The expected direct cost to prevent an attack is analyzed to determine which scenarios are cost-effective when compared to the base case. Table 5.5 reports the expected direct cost to prevent an attack for several scenarios. In all sets of scenarios with P_S and Relationship 1, 2, 3 given, the expected direct cost to prevent an attack takes its lowest values when $\alpha = 0.67$ and $\beta = 100$. This suggests that it is most cost-effective to use moderately expensive baggage screening devices for selectees with an accurate prescreening system. The expected direct cost to prevent an attack for the $\alpha = 0.67$ and $\beta = 10$ scenarios are lower than the $\alpha = 0.33$ and $\beta = 100$ scenarios for $P_S = 0.10$, 0.20 for Relationship 1 scenarios, $P_S = 0.20$ for Relationship 2 scenarios, and $P_S \geq 0.05$ for the Relationship 3 scenarios. This suggests that expensive screening devices should be used sparingly. Table 5.5 also indicates that the least cost-effective scenarios are those that employ

Table 5.5: Expected Direct Cost to Prevent an Attack ($B)

			EPA		
α	β	Relationship	$P_S = 0.05$	$P_S = 0.10$	$P_S = 0.20$
0.33	1	1	23.03	23.03	23.03
		2	8.40	8.40	8.40
		3	92.81	92.81	92.81
	10	1	3.34	4.38	6.45
		2	1.22	1.60	2.35
		3	13.46	17.63	25.99
	100	1	1.37	2.51	4.79
		2	0.50	0.92	1.75
		3	5.52	10.12	19.31
0.67	1	1	11.34	11.34	11.34
		2	5.11	5.11	5.11
		3	28.27	28.27	28.27
	10	1	1.65	2.16	3.18
		2	0.74	0.97	1.43
		3	4.10	5.37	7.92
	100	1	0.68	1.24	2.36
		2	0.30	0.56	1.06
		3	1.68	3.08	5.88

random screening.

If selective screening is used, Table 5.5 indicates that there are several scenarios that can be expected to prevent an additional attack with only a moderate increase in the direct cost (compared to the base case); these scenarios can be determined by analyzing (5.5).

Let τ denote the expected direct cost that one is willing to spend to prevent an attack. Define the *β-threshold*, $\beta^*_{\alpha,P_S}(\tau)$, as the minimal value of β (for a given P_S and $\alpha < 1$) such that the expected direct cost to prevent an attack is τ. Therefore, if $\beta > \beta^*_{\alpha,P_S}(\tau)$ for given values of $\alpha < 1$, P_S, and τ, then using selective screening is preferable to the base case of screening all baggage with EDSs.

For the special case when $\beta = +\infty$, the best-case scenario for selective screening, then $P_{\mathrm{S|T}} = 1$ and $P_{\mathrm{NS|T}} = 0$. Moreover,

$$P_{\mathrm{S|NT}} = \frac{P_S - P_{\mathrm{S|T}}P_{\mathrm{T}}}{1 - P_{\mathrm{T}}} = \frac{P_S - P_{\mathrm{T}}}{1 - P_{\mathrm{T}}}$$

and

$$P_{\mathrm{NS|NT}} = 1 - P_{\mathrm{S|NT}} = (1 - P_S)/(1 - P_{\mathrm{T}}).$$

If the expected direct cost to prevent an attack is greater than τ (when $\beta = +\infty$), then $\beta^*_{\alpha,P_S}(\tau) \equiv$

Table 5.6: Beta Threshold Values

		β^*_{α,P_S}(\$1B)			β^*_{α,P_S}(\$5B)			β^*_{α,P_S}(\$10B)		
α	P_S	R1	R2	R3	R1	R2	R3	R1	R2	R3
0.33	0.05	$+\infty$	13.8	$+\infty$	5.7	1.7	246	2.5	1.0	16.5
	0.10	$+\infty$	47.6	$+\infty$	7.7	1.8	$+\infty$	2.7	1.0	116
	0.20	$+\infty$	$+\infty$	$+\infty$	46.8	2.0	$+\infty$	3.4	1.0	$+\infty$
0.67	0.05	25	6.5	$+\infty$	2.4	1.0	7.5	1.1	1.0	3.1
	0.10	$+\infty$	9.4	$+\infty$	2.6	1.0	11.7	1.2	1.0	3.5
	0.20	$+\infty$	$+\infty$	$+\infty$	3.3	1.0	$+\infty$	1.2	1.0	5.2

$+\infty$. Otherwise,

$$\beta^*_{\alpha,P_S}(\tau) = \inf\{\beta \,|\beta > 1,\ \text{EPA} \leq \tau\} < +\infty. \tag{5.6}$$

Note that EPA is a function of β.

Table 5.6 reports values of $\beta^*_{\alpha,P_S}(\tau)$ for $\alpha = 0.33, 0.67$, $P_S = 0.05, 0.1, 0.2$, $\tau =$\$1B, \$5B, \$10B, for Relationships 1, 2, 3 (abbreviated as R1, R2, R3, respectively). To interpret the values in Table 5.6, selective screening is preferable to screening all checked baggage with EDSs if $\beta > \beta^*_{\alpha,P_S}(\tau)$ for given values of α, P_S, and τ. When $\tau =$\$1B, it is never preferable to use selective screening for the Relationship 3 scenarios, nor it is preferable to use selective screening when $P_S = 0.20$ for the Relationship 1, 2, or 3 scenarios. Moreover, for the Relationship 2 scenarios, it is preferable to use selective screening for $\beta \geq 2$ for all of the $\tau =$\$5B and \$10B scenarios, a small improvement over random screening.

5.4 Conclusion

This chapter presents risk and cost models that quantify the effectiveness of selective screening strategies that use a prescreening system, such as CAPPS II or Secure Flight. The risk model is used to measure the ability of the prescreening system to successfully recognize threat passengers as selectees. The direct cost model captures the costs associated with using two different types of screening devices for screening selectee and nonselectee baggage. It incorporates the costs associated with purchasing, operating, and maintaining baggage screening devices as a function of the number of passengers classified as selectees and nonselectees, and the direct costs associated with true clears, true alarms, and false alarms. The results quantify the expected direct cost, expected number of successful attacks per billion passengers, and the expected direct cost to prevent an attack. Some of the data needed for this are security-sensitive, and hence, were modified. However, the data used in the analysis are representative of the actual data, and the conclusions obtained from the analysis

are likely to be similar to those obtained with the actual data. The key conclusion obtained for this analysis is that the prescreening system must be accurate (i.e., $\beta > 1$) in order to reduce the number of successful attacks. The most cost-effective scenarios as compared to screening all checked baggage with EDSs occur when the prescreening system is accurate in identifying passenger risk and P_S is small.

Chapter 6

Knapsack Problems with Set-Up Weights

The classical Integer Knapsack Problem (IKP) is a well-studied problem in discrete optimization [45, 37]. IKP is defined by a set of n item types, each having a positive integer value and a positive integer weight, and a knapsack capacity. The objective of IKP is to determine the number of copies of each item type such that the sum of their weights does not exceed the capacity and the sum of their values is maximized. The 0-1 Knapsack Problem (KP) is a variation of IKP in which no more than a single copy of each item may appear in the knapsack. The Bounded Knapsack Problem (BKP) is a generalization of IKP in which an upper bound on the availability of each item type is given.

This chapter introduces and analyzes the Integer Knapsack Problem with Set-up Weights (IKPSW), a generalization of IKP in which each item type has a non-negative *set-up weight* that is added to the knapsack if any copies of that item type are in the knapsack [48]. The k-item IKPSW, denoted by (kIKPSW), is a variation of IKPSW, in which the number of items in the knapsack must be equal to a given value k. The Bounded Set-Up Knapsack Problem is an extension of IKPSW in which there is a non-negative set-up value associated with each item type and an upper bound on the availability of each item type.

IKPSW, kIKPSW, and BSKP are motivated by MAP (see Chapter 2), in which a given number of indistinguishable passengers must be screened by one of several security classes. Each security class has an overhead cost associated with purchasing and maintaining screening devices, a marginal cost associated with screening each passenger assigned to that security class, and a security value that quantifies the level of security attained when a passenger is screened in that security class. The objective is to determine the number of passengers to screen in each security class such that the budget is not exceeded and the total security is maximized. IKPSW and BSKP also model applications in the area of economics, capital budgeting, and scheduling [25].

Several aspects of IKPSW, kIKPSW, and BSKP are similar to existing problems. In particular,

set-up weights are analogous to set-up times in sequencing problems [11, 90]. The fundamental difference between these problems is that in sequencing problems, the objective is to determine a schedule for all tasks, whereas in knapsack problems, the objective is to select items for the knapsack. A k-item constraint has been analyzed for KP [13] and in conjunction with dynamic programming models for the Collapsing Knapsack Problem [64, 65]. The Bounded Knapsack Problem with Setups is a particular case of BSKP that does not include set-up values. Süral et al. [73] present a branch and bound algorithm for this problem. Since the objective function and weight constraint for BSKP are piecewise linear functions, BSKP can be formulated with nonlinear objective and weight functions. However, the objective and weight functions are neither convex nor concave. Therefore, BSKP is not a particular case of the Nonlinear Knapsack Problem, in which the objective is to maximize a separable concave objective function subject to convex packing constraints [23].

IKPSW, kIKPSW, and BSKP are particular cases of knapsack models that consider precedence constraints on the set of items. The Setup Knapsack Problem (SKP) considers partitioning the items into families that are known a priori [16]. There is a set-up item with value and weight associated with each family that must be added to the knapsack before adding any items in its family. The values and set-up values are real numbers that may be negative, rather than non-negative integers as in IKPSW, kIKPSW, and BSKP. Exact methods, including a dynamic programming algorithm, are presented for SKP. The Precedence-Constrained Knapsack Problem is a generalization of KP and SKP in which a series of precedence constraints are imposed on the variables. Ibarra and Kim [25] introduce this problem and show that it is NP-complete in the strong sense when there is an arbitrary number of precedence constraints. Park and Park [62] develop a lifting procedure of the modified cover inequality which explicitly considers the precedence constraints. Boyd [9] analyzes the polyhedral structure of the convex hull of feasible integer solutions to this problem. Johnson and Niemi [33] analyze this problem graphically and consider the particular case when the precedence constraints can be represented as a tree. They also develop a pseudo-polynomial time dynamic programming algorithm that produces exact solutions for this special case.

Although the results of Johnson and Niemi [33] can be applied to IKPSW, kIKPSW, and BSKP, the focus of this chapter is to develop algorithms and heuristics that exploit the particular structures of these three knapsack problem variations. Pseudo-polynomial time dynamic programming algorithms and fully-polynomial time approximation schemes cannot be applied to the Precedence-

Constrained Knapsack Problem since it is NP-hard in the strong sense. To see that IKPSW is a particular instance of the Precedence-Constrained Knapsack Problem, first transform the IKPSW instance into a KP instance by ignoring the set-up weights. For each item type, add an item whose weight is the sum of weight of the item type and the set-up weight, and obtain a set of precedence constraints such that the item with the set-up weight must be added before the other items of that type. BSKP can be shown to be a particular case of the Precedence-Constrained Knapsack Problem by using an analogous transformation.

IKPSW can also be transformed into a particular case of the Multiple-Choice Knapsack Problem (MCKP) [37]. MCKP is a more general knapsack problem variation, which allows a more flexible handling of the set-up weights. In MCKP, the set of items are partitioned into classes, and exactly one item from each class must be added to the knapsack. To transform IKPSW into an instance of MCKP, a class is created for each item type. In each class, a set of items are created to account for all possible multiplicities of the item type of interest (i.e., the MCKP value is a multiple of the value, and the MCKP weight is the sum of the set-up weight and a multiple of the weight), including an item with weight and value equal to zero. The disadvantage of the MCKP formulation is that it creates a large number of items. Kellerer et al. [37] summarize several algorithms and heuristics for MCKP.

This chapter is organized as follows. Section 6.1 formulates IKPSW, kIKPSW, and BSKP as discrete optimization problems and integer programs. Section 6.2 describes two pseudo-polynomial time dynamic programming algorithms for solving IKPSW, and presents two approximation algorithms for IKPSW, including a Greedy heuristic and a fully polynomial time approximation scheme (FPTAS), which produce solutions to IKPSW that are within a factor of 1/2 and ε of the optimal solution value, respectively, with $\varepsilon \leq 1/2$. Section 6.3 extends the algorithms and heuristics to kIKPSW. Section 6.4 presents pseudo-polynomial dynamic programming algorithms for BSKP, a Greedy heuristic, and a FPTAS for BSKP. Section 6.5 provides concluding comments.

The following terminology is used throughout this chapter. Let Π be an optimization problem with optimal value z. Define heuristic H to be a ε-approximation algorithm for Π, $\varepsilon \in [0, 1]$, if and only if

$$\frac{z - z^h}{z} \leq \varepsilon$$

for all instances of Π where z^h is the value determined by H. The value ε is referred to as the relative error bound. A FPTAS takes as input an approximation ratio $\varepsilon \in (0, 1)$ and has a running

time polynomial in the length of the input and in $1/\varepsilon$ [61].

6.1 Preliminaries

This section formulates IKPSW, kIKPSW, and BSKP as discrete optimization models and integer programs. In addition, all three of these knapsack variations are shown to be NP-hard. IKPSW is formally stated.

Integer Knapsack Problem with Set-up Weights (IKPSW)
Given a set of n item types, values $v_i \in Z_0^+$ corresponding to each item type, $i = 1, 2, \ldots, n$, weights $w_i \in Z^+$ corresponding to each item type, $i = 1, 2, \ldots, n$, set-up weights $s_i \in Z_0^+$ corresponding to each item type $i = 1, 2, \ldots, n$, and capacity $c \in Z^+$, find non-negative integers $x_1, x_2, \ldots, x_n$, such that $\sum_{i=1}^{n} w_i x_i + \sum_{i:x_i>0} s_i \leq c$ and $\sum_{i=1}^{n} v_i x_i$ is maximized.

Without loss of generality, assume that $s_i + w_i \leq c$, $i = 1, 2, \ldots, n$ (i.e., at least one copy of each item type can fit in the knapsack). IKPSW is NP-hard since it is a generalization of IKP, which itself is NP-hard [42].

Theorem 6.1 *IKPSW is NP-Hard*

Proof. IKP is a particular case of IKPSW with $s_i = 0$, $i = 1, 2, \ldots, n$. □

IKPSW can be formulated as an integer programming (IP) model (see (6.1)-(6.4)), where the integer decision variables x_i indicate how many items of type i are added to the knapsack, $i = 1, 2, \ldots, n$, and the binary decision variables y_i indicate if the set-up weight of item type i is or is not assessed to the knapsack, $i = 1, 2, \ldots, n$.

$$\max \sum_{i=1}^{n} v_i x_i \tag{6.1}$$

$$\text{subject to } \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} s_i y_i \leq c \tag{6.2}$$

$$\frac{x_i}{B} - y_i \leq 0, i = 1, 2, \ldots, n \tag{6.3}$$

$$x_i \in Z_0^+, i = 1, 2, \ldots, n;\ y_i \in \{0, 1\}, i = 1, 2, \ldots, n \tag{6.4}$$

Define $B = \max_{i=1,2,\ldots,n} \left\{ \left\lfloor \frac{c-s_i}{w_i} \right\rfloor \right\}$. In (6.1), the objective of IKPSW is to maximize the total value of the items in the knapsack. The first constraint, (6.2), ensures that the weight of the items in

the knapsack, including their set-up weights, do not exceed the capacity. In the second set of n constraints, (6.3), $y_i = 1$ if any items of type i are in the knapsack (i.e., $x_i > 0$). The value of B is a scale factor such that $0 \leq x_i/B \leq 1$, $i = 1, 2, \ldots, n$. The final set of $2n$ constraints, (6.4), indicate that $x_1, x_2, \ldots, x_n$ are non-negative integer variables and $y_1, y_2, \ldots, y_n$ are binary variables.

The k-item Integer Knapsack Problem with Set-up Weights, kIKPSW, is a variation of IKPSW in which exactly k items must be added to the knapsack. This restriction is enforced by a k-item constraint, called the *cardinality constraint*. Without loss of generality, assume that for every item type i, there exists a feasible solution to kIKPSW that uses at least one copy of type i, $i = 1, 2, \ldots .n$. kIKPSW is formally stated, and it is shown to be NP-hard.

k-item Integer Knapsack Problem with Set-up Weights (kIKPSW)
Given n, v, w, s, and c as defined in IKPSW, and cardinality $k \in Z^+$, find non-negative integers $x_1, x_2, \ldots, x_n$, such that $\sum_{i=1}^{n} x_i = k$, $\sum_{i=1}^{n} w_i x_i + \sum_{i:x_i>0} s_i \leq c$, and $\sum_{i=1}^{n} v_i x_i$ is maximized.

Theorem 6.2 *kIKPSW is NP-hard.*

Proof. The Exact k-item Knapsack Problem is a variation of KP where the number of items in a feasible solution must be equal to k [13]. The Exact k-item Knapsack Problem is a particular case of kIKPSW when $s_i = 0,\ i = 1, 2, \ldots, n$. □

The IP formulation for kIKPSW is the same as the IP model for IKPSW, with $B = k$ in (6.3), and with the addition of a k-item cardinality constraint,

$$\sum_{i=1}^{n} x_i = k. \tag{6.5}$$

The variables are also identical to those in the IKPSW.

kIKPSW is a particular case of MAP in which passengers are indistinguishable (see Chapter 2) [52]. The objective of MAP is to determine how many passengers should be screened by each security class such that all N passengers are screened, the cost to screen the N passengers is within budget, and the security is maximized. Note that this particular case of MAP is defined as an instance kIKPSW with $N = k$, $M = n$, $B = c$, fixed costs equal to the set-up weights, marginal costs equal to the weights, and security values equal to the values (the security levels can be rescaled to be between zero and one).

BSKP generalizes IKPSW by including a bound for the availability of each item type and a set-up value associated with each item type that is added to the objective value if any copies of

that item type are added to the knapsack. BSKP is formally stated.

The Bounded Set-up Knapsack Problem (BSKP)

Given n, v, w, s, and c as defined in IKPSW, set-up values $u_i \in Z_0^+$, and bounds $b_i \in Z^+$ corresponding to each item type $i = 1, 2, \ldots, n$, find non-negative integers $x_1, x_2, \ldots, x_n$, such that $\sum_{i=1}^{n} w_i x_i + \sum_{i:x_i>0} s_i \leq c$, $x_i \leq b_i$, $i = 1, 2, \ldots, n$, and $\sum_{i=1}^{n} v_i x_i + \sum_{i:x_i>0} u_i$ is maximized.

Without loss of generality, assume that $s_i + b_i w_i \leq c$, $i = 1, 2, \ldots, n$ (i.e., the bounds are defined such that b_i copies of item type $i = 1, 2, \ldots, n$ can fit into the knapsack) and that $\sum_{i=1}^{n}(s_i + b_i w_i) > c$ to ensure a nontrivial solution. Note that this implies that $b_i \leq \lfloor (c - s_i)/w_i \rfloor$, $i = 1, 2, \ldots, n$. BSKP is trivially NP-hard since it is a generalization of IKPSW, which itself is NP-hard [48].

Theorem 6.3 *BSKP is NP-hard.*

Proof. IKPSW is a particular case of BSKP with $u_i = 0$, $i = 1, 2, \ldots, n$, and $b_i = \lfloor (c - s_i)/w_i \rfloor$, $i = 1, 2, \ldots, n$. □

BSKP can be formulated as an integer programming (IP) model (see (6.6)—(6.11)), where the integer decision variables x_i indicate how many items of type $i = 1, 2, \ldots, n$ are added to the knapsack, and the binary decision variables y_i indicate if any copies of item type $i = 1, 2, \ldots, n$ are in the knapsack.

$$\max \sum_{i=1}^{n} v_i x_i + \sum_{i=1}^{n} u_i y_i \tag{6.6}$$

$$\text{subject to } \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} s_i y_i \leq c \tag{6.7}$$

$$\frac{1}{b_i} x_i - y_i \leq 0, \ i = 1, 2, \ldots, n \tag{6.8}$$

$$y_i \leq x_i, \ i = 1, 2, \ldots, n \tag{6.9}$$

$$x_i \in \{0, 1, \ldots, b_i\}, \ i = 1, 2, \ldots, n, \tag{6.10}$$

$$y_i \in \{0, 1\}, \ i = 1, 2, \ldots, n. \tag{6.11}$$

In (6.6), the objective of BSKP is to maximize the total value of the items in the knapsack, including set-up values. The first constraint, (6.7), ensures that the weight of the items in the knapsack, including their set-up weights, do not exceed the knapsack capacity. The second and third sets of $2n$ constraints, (6.8) and (6.9), guarantee that $y_i = 1$ if any items of type i are in the knapsack (i.e., $x_i > 0$) and that $y_i = 0$ otherwise, $i = 1, 2, \ldots, n$. The fourth set of n constraints, (6.10),

indicates that x_i is a non-negative integer within its bounds, and the final set of n constraints, (6.11), indicates that y_i is binary, $i = 1, 2, \ldots, n$.

6.2 IKPSW

This section provides two dynamic programming algorithms, a Greedy heuristic, and a FPTAS for IKPSW.

6.2.1 Dynamic Programming Algorithms

This section introduces two dynamic programming algorithms for IKPSW. To describe the first dynamic programming algorithm (labeled DP-V), let $z_r(d)$ be the optimal solution value to the following knapsack subproblem defined on the first $r = 1, 2, \ldots, n$ item types with value $d = 1, 2, ..., U$, where U is an upper bound on the optimal solution value. Then,

$$z_r(d) = \min \left\{ \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r} s_i y_i \left| \sum_{i=1}^{r} v_i x_i \geq d,\ \frac{1}{B} x_i \leq y_i,\ i = 1, 2, \ldots, r \right. \right\},$$

with decision variables $x_i \in Z_0^+$ and $y_i \in \{0, 1\},\ i = 1, 2, ..., n$. By design, the optimal solution value of IKPSW is given by the largest value of d such that $z_n(d) \leq c$.

Let $\bar{z}_r(d)$ be the optimal solution value over the first $r = 1, 2, \ldots, n$ item types with value $d = 0, 1, ..., U$, given that at least one item of type r is present in the knapsack,

$$\bar{z}_r(d) = \min \left\{ \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r-1} s_i y_i + s_r \left| \sum_{i=1}^{r} v_i x_i \geq d;\ \frac{1}{B} x_i - y_i \leq 0 \right. \right.$$
$$\left. i = 1, 2, ..., r-1;\ x_r \geq 1 \right\}.$$

Initially, $z_r(0) = 0$, $r = 1, 2, \ldots, n$, and $\bar{z}_r(0) = +\infty$, $i = 1, 2, \ldots, n$. In addition, $z_1(d) = \bar{z}_1(d) = s_1 + w_1 \lceil \frac{d}{v_1} \rceil$, $d = 0, 1, \ldots, U$.

Subsequent values of $z_r(d)$ are obtained by the recursion

$$z_r(d) = \min\{z_{r-1}(d),\ \bar{z}_r(d)\}, \tag{6.12}$$

while subsequent values of $\bar{z}_r(d)$ are obtained from the recursion

$$\bar{z}_r(d) = \min\{z_{r-1}(d - v_r) + s_r + w_r,\ \bar{z}_r(d - v_r) + w_r\}.$$

Let $\bar{z}_r(d) = +\infty$ if no solution exists. In this way, $\bar{z}_r(d)$ and $z_r(d)$ work together to identify the optimal solution. For each $z_r(d),\ d = 0, 1, \ldots, U$, the index of the last item type added to the knapsack is recorded. The optimal solution can be found by backtracking through these values for $r = n,\ d = 0, 1, \ldots, U$ [37].

DP-V finds an optimal solution to IKPSW in $O(nU)$ time since each step in the recursion calls at most two other recursions. DP-V requires $O(n + U)$ space since only the entries $\bar{z}_r(d)$ and $z_{r-1}(d),\ d = 0, 1, \ldots, U$, are needed to determine $z_r(d),\ r = 1, 2, \ldots, n$ [37]. DP-V is described in pseudocode by Algorithm 3. Since Algorithm 3 reflects the storage reduction scheme, $z(d)$ and $\bar{z}(d)$ represent $z_r(d)$ and $\bar{z}_r(d)$, respectively. Let $l(d)$ denote the index of the last item added to the knapsack in $z(d)$, $f(d) = 1(0)$ if this is (not) the first copy of item type $l(d)$ added to the knapsack in $z(d)$, and Let $\bar{f}(d) = 1(0)$ if this is (not) the first copy of the item type added to the knapsack in $\bar{z}(d)$.

To describe the second dynamic programming model (labeled DP-W), let $z_r(\bar{c})$, $r = 1, 2, \ldots, n$, $\bar{c} = 0, 1, \ldots, c$, be the optimal solution value to the following knapsack subproblem defined on the first r item types:

$$z_r(\bar{c}) = \max\left\{ \sum_{i=1}^{r} v_i x_i \,\middle|\, \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r} s_i y_i \le \bar{c};\ \frac{1}{B} x_i - y_i \le 0,\ i = 1, 2, ..., r \right\},$$

with $B = \max_{i=1,2,\ldots,n}\left\{ \lfloor \frac{c - s_i}{w_i} \rfloor \right\}$, decision variables $x_i \in Z_0^+$ and $y_i \in \{0, 1\},\ i = 1, 2, ..., n$. By design, the optimal solution value of IKPSW is given by $z_n(c)$. Let $\bar{z}_r(\bar{c}),\ r = 1, 2, \ldots, n,\ \bar{c} = 1, 2, \ldots, c$, be the optimal solution value over the first r item types with capacity $\bar{c}$, given that at least one item of type r is present in the knapsack,

$$\bar{z}_r(\bar{c}) = \max\left\{ \sum_{i=1}^{r} v_i x_i \,\middle|\, \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r-1} s_i y_i + s_r \le \bar{c};\ \frac{1}{B} x_i - y_i \le 0 \right.$$
$$\left. i = 1, 2, ..., r-1;\ x_r \ge 1 \right\}.$$

Initially, $\bar{z}_1(\bar{c}) = z_1(\bar{c}) = 0$ for $\bar{c} < s_1 + w_1$ and $\bar{z}_1(\bar{c}) = z_1(\bar{c}) = v_1 \left\lfloor \frac{\bar{c} - s_1}{w_1} \right\rfloor$ for $\bar{c} \ge s_1 + w_1$. Subsequent values of $\bar{z}_r(\bar{c})$ are obtained from the recursion

$$\bar{z}_r(\bar{c}) = \max\{ z_{r-1}(\bar{c} - s_r - w_r) + v_r,\ \bar{z}_r(\bar{c} - w_r) + v_r \}$$

Algorithm 3 IKPSW Dynamic Programming Algorithm DP-V

$z(0) \leftarrow 0,\ l(0) \leftarrow f(0) \leftarrow 0,\ \bar{z}(0) \leftarrow \infty,\ \bar{x}_r(0) \leftarrow 0$
for $d \leftarrow 1$ to U **do**
  Comment: Set boundary conditions
  $z(d) \leftarrow \bar{z}(d) \leftarrow s_1 + w_1\lceil d/v_1 \rceil(d),\ l(d) \leftarrow 1$
end for
for $d \leftarrow 1$ to $v_1 + u_1$ **do**
  $f(d) \leftarrow 1,\ \bar{f}(d) \leftarrow 1$
end for
for $r \leftarrow 2$ to n **do**
  for $d \leftarrow 1$ to U **do**
    Comment: Update $\bar{z}(d)$
    $\bar{d} \leftarrow \max\{0,\ d - v_r\},$
    if $z(\bar{d}) + s_r + w_r \leq \bar{z}(\bar{d}) + w_r$ **then**
      $\bar{z}(d) \leftarrow z(\bar{d}) + s_r + w_r,\ \bar{f}(d) \leftarrow 1$
    else
      $\bar{z}(d) \leftarrow \bar{z}(\bar{d}) + w_r,\ \bar{f}(d) \leftarrow 0$
    end if
  end for
  for $d \leftarrow 1$ to U **do**
    Comment: Update $z(d)$
    if $\bar{z}(d) < z(d)$ **then**
      $z(d) \leftarrow \bar{z}(d),\ l(d) \leftarrow r,\ f(d) \leftarrow \bar{f}(d)$
    end if
  end for
end for
$z^{OPT} \leftarrow \max\{d | z(d) \leq c\}$
Comment: Obtain X^{OPT}
$d \leftarrow z^{OPT},\ x_r^{OPT} \leftarrow 0,\ r = 1, 2, \ldots, n$
while $d > 0$ **do**
  $x_{l(d)}^{OPT} \leftarrow x_{l(d)}^{OPT} + 1$
  $d \leftarrow d - v_{l(d)}$
end while
return $z^{OPT},\ X^{OPT}$

($\bar{z}_r(\bar{c}) = -\infty$ if $\bar{c} < s_r + w_r$) while subsequent values of $z_r(\bar{c})$ are obtained from the recursion

$$z_r(\bar{c}) = \max\{z_{r-1}(\bar{c}),\ \bar{z}_r(\bar{c})\}. \tag{6.13}$$

At each step of the recursion, either no items of type r are added to the knapsack and $z_{r-1}(\bar{c})$ is called, or a copy of item type r is added to the knapsack and $\bar{z}_r(\bar{c})$ is called. If $\bar{z}_r(\bar{c})$ is called, exactly one item of type r is added to the knapsack and $z_{r-1}(\bar{c} - s_r - w_r)$ is called, or at least one item of type r is added to the knapsack and $\bar{z}_r(\bar{c} - w_r)$ is called. In this way, $\bar{z}_r(\bar{c})$ and $z_r(\bar{c})$ work together to identify the optimal solution.

DP-W finds an optimal solution to IKPSW in $O(nc)$ time since each step in the recursion calls at most two other recursions, and requires $O(n + c)$ space since only the entries $\bar{z}_r(\bar{c})$ and $z_{r-1}(\bar{c})$, $\bar{c} = 1, 2, \ldots, c$, are needed to determine $z_r(\bar{c})$, $r = 1, 2, \ldots, n$.

DP-W is described in pseudocode by Algorithm 4. Since Algorithm 4 reflects the storage reduction scheme, $z(\bar{c})$ and $\bar{z}(\bar{c})$ represent $z_r(\bar{c})$ and $\bar{z}_r(\bar{c})$, respectively. Let $l(\bar{c})$ denote the index of the last item added to the knapsack in $z(\bar{c})$, $f(\bar{c}) = 1(0)$ if this is (not) the first copy of item type $l(\bar{c})$ added to the knapsack in $z(\bar{c})$, and Let $\bar{f}(\bar{c}) = 1(0)$ if this is (not) the first copy of the item type added to the knapsack in $\bar{z}(\bar{c})$.

6.2.2 Heuristics

This section presents two heuristics for IKPSW, a Greedy heuristic and a FPTAS. The Greedy heuristic produces solutions that are within a factor of 1/2 of the optimal solution. An example shows that this 1/2 relative error bound is tight. The FPTAS uses the Greedy heuristic to produce solutions whose values are within a factor of ε of the optimal solution value, given a relative error bound $\varepsilon \leq 1/2$, in a running time that is polynomial in the size of the encoded problem instance and in $1/\varepsilon$.

The first heuristic, the IKPSW Greedy heuristic (labeled $H^{1/2}$) considers the partial continuous relaxation of the variables $x_1, x_2, \ldots, x_n$ in which a fractional number of copies of any item type may be added to the knapsack, while the variables $y_1, y_2, \ldots, y_n$ remain binary. An approximate integer feasible solution to IKPSW can be constructed by adding the item type that will result in the largest value of the partial continuous relaxation. Define the remaining capacity in the knapsack as $\bar{c}$, with $0 \leq \bar{c} \leq c$. Then, a function that returns the largest value using only item type

Algorithm 4 IKPSW Dynamic Programming Algorithm DP-W

set $z(\bar{c}) \leftarrow 0$, $\bar{z}(\bar{c}) \leftarrow -\infty$, $l(\bar{c}) \leftarrow 0$, $\bar{c} = 0, 1, \ldots, c$
for $\bar{c} \leftarrow w_1 + s_1$ to c **do**
  Comment: Set boundary conditions
  $z(\bar{c}) \leftarrow v_1 \lfloor (\bar{c} - s_1)/w_1 \rfloor$, $l(\bar{c}) \leftarrow 1$
  if $\lfloor (\bar{c} - s_1)/w_1 \rfloor > 0$ **then**
    $\bar{z}(\bar{c}) \leftarrow z(\bar{c})$
  end if
end for
for $\bar{c} \leftarrow w_1 + s_1$ to $2w_1 + s_1 - 1$ **do**
  $f(\bar{c}) \leftarrow 1$, $\bar{f}(\bar{c}) \leftarrow 1$
end for
for $r \leftarrow 2$ to n **do**
  for $\bar{c} \leftarrow 0$ to $s_r + w_r - 1$ **do**
    $\bar{z}(\bar{c}) \leftarrow -\infty$, $\bar{f}(\bar{c}) \leftarrow 0$
  end for
  for $\bar{c} \leftarrow s_r + w_r$ to c **do**
    Comment: Update $\bar{z}_r(\bar{c})$
    if $z(\bar{c} - s_r - w_r) + v_r \geq \bar{z}(\bar{c} - w_r) + v_r$ **then**
      $\bar{z}(\bar{c}) \leftarrow z_{r-1}(\bar{c} - s_r - w_r) + v_r$, $\bar{f}(\bar{c}) \leftarrow 1$
    else
      $\bar{z}(\bar{c}) \leftarrow \bar{z}(\bar{c} - w_r) + v_r$
    end if
  end for
  for $\bar{c} \leftarrow s_r + w_r$ to c **do**
    Comment: Update $z_r(\bar{c})$
    if $z(\bar{c}) < \bar{z}(\bar{c})$ **then**
      $z(\bar{c}) \leftarrow \bar{z}(\bar{c})$, $l(\bar{c}) \leftarrow r, f(\bar{c}) \leftarrow \bar{f}(\bar{c})$
    end if
  end for
end for
$\bar{c} \leftarrow c$, $x_r^{OPT} \leftarrow 0$, $r = 1, 2, \ldots, n$
while $\bar{c} > 0$ **do**
  $x_{l(\bar{c})}^{OPT} \leftarrow x_{l(\bar{c})}^{OPT} + 1$
  if $f(\bar{c}) > 0$ **then**
    $\bar{c} \leftarrow \bar{c} - w_{l(\bar{c})} - s_{l(\bar{c})}$
  else
    $\bar{c} \leftarrow \bar{c} - w_{l(\bar{c})}$
  end if
end while
return $z^{OPT} \leftarrow z(\bar{c})$, x^{OPT}

$i = 1, 2, \ldots, n$ given capacity $\bar{c}$ and allowing a fractional number of copies is defined as

$$g_i(\bar{c}) = \max\left\{\frac{\bar{c} - s_i}{w_i} v_i,\ 0\right\}.$$

Note that each $g_i(\bar{c})$ is a continuous, non-decreasing, piecewise linear function with no more than two linear components $0 \leq \bar{c} \leq c$.

The function $g^*(\bar{c}) = \max_i\{g_i(\bar{c})\}$ is defined for $0 \leq \bar{c} \leq c$ as the optimal value of the partial continuous relaxation of IKPSW given $\bar{c}$. Note that this solution requires a single item type. Constructing $g^*(\bar{c})$ for $0 \leq \bar{c} \leq c$ requires $O(n \log n)$ time by using line intersection algorithms [8]. Once defined, $g^*(\bar{c})$ is a continuous, non-decreasing, piecewise linear function composed of no more than n linear components that returns the optimal value to the partial continuous relaxation in constant time.

An integer feasible solution is constructed as follows. Let $i^* = \arg\max g^*(\bar{c})$. Starting with $\bar{c} = c$, $\lfloor(\bar{c} - s_{i^*})/w_{i^*}\rfloor$ copies of item type i^* can be added to knapsack, with $\bar{c}$ then adjusted. This process is repeated until either $\bar{c} = 0$, $g^*(\bar{c}) = 0$, or i^* is unchanged for two successive iterations. Note that an upper bound for the optimal value of IKPSW is

$$z^u = \left\lfloor \frac{c - s_{i^*}}{w_{i^*}} v_{i^*} \right\rfloor,$$

which contains only one item type. $H^{1/2}$ requires time and space bounds of $O(n \log n)$ and $O(n)$, respectively. Theorem 6.4 shows that $H^{1/2}$ always produces a solution within a factor of $1/2$ of the optimal solution.

Theorem 6.4 *$H^{1/2}$ is a 1/2-approximation algorithm for IKPSW.*

Proof. Without loss of generality, assume that $i^* = 1$. Let $z^g = v_1\lfloor(c - s_1)/w_1\rfloor$, the best integer solution value when only item type 1 is added to the knapsack. Then,

$$z^g \leq z^h \leq z^u = \left\lfloor \frac{c - s_1}{w_1} v_1 \right\rfloor \leq v_1 \left(\left\lfloor \frac{c - s_1}{w_1} \right\rfloor + 1 \right) \leq 2z^g$$

□

Example 1 shows that the relative error bound of $1/2$ given in Theorem 6.4 for $H^{1/2}$ is tight.

Example 1: Consider the series of examples with $n = 2$, $w_1 = w_2 = v_2 = r$, $v_1 = r + 1$, $s_1 = 1$, $s_2 = 0$, and $c = 2r$. The heuristic solution value is $r + 1$ since it adds one copy of item type 1, and

the optimal solution value is $2r$ since it adds two copies of item type 2. The bound is tight since $z^h/z = (r+1)/r$ is arbitrarily close to 1/2 for r sufficiently large. □

The relative error bound given by Theorem 6.4 applies to any arbitrary instance of IKPSW. Corollary 6.1 gives a more predictive relative error bound for any particular instance of IKPSW by using the size, size overhead, and capacity parameters, where a worst-case relative error bound of 1/2 corresponds to $l = 1$. Corollary 6.1 also shows that half the upper bound found by $H^{1/2}$ provides a lower bound for the optimal solution of IKPSW, and that the absolute error of the solution produced by $H^{1/2}$ is bounded above by the largest value. The absolute error is defined as the difference between the optimal and heuristic solutions (i.e., $z - z^h$).

Corollary 6.1 *The following three results hold for $H^{1/2}$:*

(a) $H^{1/2}$ produces solutions to IKPSW such that $z^h \geq \frac{l}{l+1} z$, where $l = \min_{i=1,2,\ldots,n}\{\lfloor (c-s_i)/w_i \rfloor\}$,

(b) half of the upper bound provides a lower bound for the optimal solution (i.e., $z^u/2 \leq z$),

(c) the absolute error is no greater than $v_{max} = \max_{i=1,2,\ldots,n}\{v_i\}$

Proof. (a) Follows immediately from the proof of Theorem 6.4, since

$$\frac{z^h}{z} \geq \frac{\left\lfloor \frac{c-s_1}{w_1} \right\rfloor}{\frac{c-s_1}{w_1}} \geq \frac{\left\lfloor \frac{c-s_1}{w_1} \right\rfloor}{\left\lfloor \frac{c-s_1}{w_1} \right\rfloor + 1} \geq \frac{l}{l+1}.$$

(b) Follows directly from $z^u \leq 2z^h$.

(c) The absolute error is bounded by $\left(\frac{c-s_1}{w_1} - \lfloor \frac{c-s_1}{w_1} \rfloor\right) v_1 \leq v_1 \leq v_{max}$. □

The IKPSW approximation scheme, $F(\varepsilon)$, uses $H^{1/2}$ and DP-V to find solutions to IKPSW that are within a factor of ε of the optimal solution. There are three phases for $F(\varepsilon)$ with $\varepsilon < 1/2$, a preprocessing phase, a dynamic programming phase, and a Greedy phase.

The first phase of $F(\varepsilon)$ obtains the set of large item types and the set of small item types to be used in the dynamic programming and Greedy phases, respectively. In typical FPTASs for KP and its variations, each item type is exclusively large or small, and copies of large item types are added to the knapsack one at a time in the dynamic programming phase. In $F(\varepsilon)$, each item type may be both large and small, and multiple copies of the large item types are added to the knapsack in the dynamic programming phase. This allows the knapsack to be coarsely filled in the dynamic programming phase, while it is filled by single copies of items in the Greedy phase.

First, $H^{1/2}$ is called to a find lower bound on the optimal solution value, z^h. Denote the threshold value by θ with $\theta = \frac{1}{2}\varepsilon z^h$. Let q_i be the smallest positive integer such that $q_i v_i \geq \theta$. Item type $i = 1, 2, \ldots, n$, is *large* if adding q_i copies forms a feasible solution (i.e., $s_i + w_i q_i \leq c$). Denote the set of large items by $L \subseteq \{1, 2, \ldots, n\}$. Item type $i = 1, 2, \ldots, n$, is *small* if $v_i < \theta$. Denote the set of small items by $S \subseteq \{1, 2, \ldots, n\}$. Note that L and S are not mutually exclusive subsets of $\{1, 2, \ldots, n\}$ unless $q_i = 1$, $i \in L$. The preprocessing phase requires $\mathrm{O}(n \log n)$ time and $\mathrm{O}(n)$ space.

The second phase of $F(\varepsilon)$ is the dynamic programming phase, in which IKPSW is first solved with the large item types with DP-V. First, the values of the large item types are scaled by a factor of $\delta = \frac{1}{4}\varepsilon^2 z^h$. Second, instead of adding a single copy of item type $i \in L$ at an iteration of DP-V, q_i copies of item type i are added to the knapsack during a single iteration. Therefore, each large item type $i \in L$ is described as having weight $\bar{w}_i = q_i w_i$ and value $\bar{v}_i = \lfloor v_i q_i / \delta \rfloor$. Moreover, the number of scaled values considered are bounded above by $8/\varepsilon^2$. The dynamic programming phase requires $\mathrm{O}(n/\varepsilon^2)$ time and $\mathrm{O}(n + 1/\varepsilon^2)$ space.

The third phase of $F(\varepsilon)$ is the Greedy phase, where the small item types are inserted into the dynamic programming solutions using $H^{1/2}$ with the remaining capacity. The Greedy phase requires $\mathrm{O}(1/\varepsilon^3)$ time and $\mathrm{O}(n)$ space. Note that the absolute and relative error bounds given by Theorem 6.4 and Corollary 6.1 require only one item type to be added to the knapsack. Finding this item—labeled the Greedy item—is achieved by considering two cases: the small item types already in the knapsack solution, and the rest of the small item types. Since the set-up weights of some of the small items may already be in the knapsack, these set-up weights must be adjusted to zero. First, the solution set in each dynamic programming solution is determined, which requires $\mathrm{O}(1/\varepsilon)$ since there are at most $4/\varepsilon$ large items in any such solution set. Define item type i' as the item type already present in the knapsack solution with the largest value to weight ratio (i.e., $v_{i'}/w_{i'}$ is maximized). Note that the Greedy item does not depend on $\bar{c}$, $0 \leq \bar{c} \leq c$, if it is already in the knapsack solution. Second, assuming that no set-up weights are in the knapsack, then item type $i^* = \arg\max\{g^*(\bar{c})\}$. Note that $g^*(\bar{c})$ is determined in the preprocessing phase. If $g^*(\bar{c}) > v_{i'}(\bar{c}/w_{i'})$, then item type i^* is added to the knapsack solution. Otherwise, item type i' is added to the knapsack solution. Note that item type i^* may already be in the solution set of the dynamic programming solution, and hence, the set-up weight is already in the knapsack. In this case, the improvement in the solution value is at least as large relative to when the set-up weight

has not been added to the knapsack.

The total time bound for F(ε) is O($n \log n + n/\varepsilon^2 + 1/\varepsilon^3$), and the total space bound for $F(\varepsilon)$ is O($n + 1/\varepsilon^2$). These bounds may be improved by using other techniques [41, 37]. Theorem 6.5 shows that the solution produced by $F(\varepsilon)$ is within a factor of ε of the optimal solution.

Theorem 6.5 *$F(\varepsilon)$ is an FPTAS for IKPSW.*

Proof. Let z^f be the value of the solution produced by $F(\varepsilon)$ and z be the optimal solution value. Error between the heuristic and optimal solutions accumulates in the dynamic programming and Greedy phases. For item type $i \in L$ in the dynamic programming phase,

$$\delta \bar{v}_i \leq q_i v_i < \delta(\bar{v}_i + 1).$$

Since at most z/θ large items can fit in any dynamic programming solutions, then the error during the dynamic programming phase is limited by $\delta z/\theta$. The error in the Greedy phase is given by $\max_{i \in S} v_i \leq \theta$ (see Corollary 6.1c). Therefore, the relative error is bounded above by ε since

$$\frac{z - z^f}{z} \leq \frac{\delta}{\theta} + \frac{\theta}{z} = \frac{\varepsilon^2 z^h/4}{\varepsilon z^h/2} + \frac{\varepsilon z^h/2}{z} \leq \varepsilon.$$

□

6.3 IKPSW with a Cardinality Constraint

This section analyzes kIKPSW, the k-item Integer Knapsack Problem with Set-up Weights. A dynamic programming algorithm and two heuristics, including a Greedy heuristic and a FPTAS, are formulated and analyzed for kIKPSW. Since these algorithms and heuristics are analogous to those demonstrated for IKPSW, brief descriptions are given unless more explanation is required.

The first dynamic programming algorithm (labeled kDP-V) is an extension of DP-V. It is similar to the approach used for the Collapsing Knapsack Problem [64]. Let $z_{r,j}(d)$ be the minimum weight solution value over the first $r = 1, 2, \ldots, n$ item types with cardinality $j = 1, 2, \ldots, k$ and with value at least $d = 0, 1, \ldots, U$. Let $\bar{z}_{r,j}(d)$ be the minimum weight solution value over the first $r = 1, 2, \ldots, n$ item types with cardinality $j = 1, 2, \ldots, k$ and with value at least $d = 0, 1, \ldots, U$, with at least one copy of item type r in the knapsack solution. Subsequent values of $\bar{z}_{r,j}(d)$ are obtained from the

recursion

$$\bar{z}_{r,j}(d) = \min\{z_{r-1,j-1}(d - v_r) + w_r + s_r,\ \bar{z}_{r,j-1}(d - v_r) + w_r\},$$

and subsequent values of $z_{r,j}(d)$ are obtained from the recursion

$$z_{r,j}(d) = \min\{z_{r-1,j}(d),\ \bar{z}_{r,j}(d)\}.$$

For each $z_{r,j}(d),\ d = 0, 1, \ldots, U,\ j = 1, 2, \ldots, k$, the index of the last item type added to the knapsack is recorded. The optimal solution can be found by backtracking through these values for $r = n,\ j = 1, 2, \ldots, k,\ d = 0, 1, \ldots, U$ as done by DP-V. An optimal solution to kIKPSW is found in $O(knU)$ time and $O(n + kU)$ space. A second dynamic programming algorithm (labeled kDP-W) extends DP-W in a similar way, which finds optimal solutions to kIKPSW in $O(knc)$ time and $O(n + kc)$ space

The kIKPSW Greedy heuristic, $H_k^{1/2}$, produces solutions to kIKPSW that use two item types and are within a factor of 1/2 of the optimal solution. There are three steps to $H_k^{1/2}$,

1. Find the best solution that fills the knapsack with a single item type.

2. Find the best solution that fills the knapsack with exactly two item types.

3. Return the best solution found in steps 1 and 2.

To obtain an approximate solution to kIKPSW, $H_k^{1/2}$ partitions the set of items into two groups: $S^{k^-} = \{i : \lfloor (c - s_i)/w_i \rfloor < k\}$, and $S^{k^+} = \{i : \lfloor (c - s_i)/w_i \rfloor \geq k\}$, where S^{k^-} and S^{k^+} are mutually exclusive and exhaustive subsets of $\{1, 2, \ldots, n\}$. Denote $n^+ = |S^{k^+}|$ and $n^- = |S^{k^-}|$.

The *limited relaxation* of kIKPSW is the partial continuous relaxation of kIKPSW in which the variables $x_1, x_2, \ldots, x_n$ in the associated integer program can take on real values, while the variables $y_1, y_2, \ldots, y_n$ remain binary. The optimal limited relaxation value provides an upper bound to the optimal solution to kIKPSW. Lemma 6.1 shows that the optimal limited relaxation of kIKPSW uses no more than two item types.

Lemma 6.1 *The optimal solution to the limited relaxation of kIKPSW uses no more than two item types. Therefore, the optimal solution of the limited relaxation of kIKPSW is a 1-item type or 2-item type solution.*

Proof. The linear programming solution to the limited relaxation has two constraints, and hence, it has at most two basic (nonzero) variables, whereas the remaining variables are zero. □

The optimal solution to the limited relaxation can be found by considering the *b-item type restricted kIKPSW*, a limited relaxation that uses a subset of the item types. Furthermore, the set-up weights of these item types are initially added to the knapsack. Therefore, there are no binary variables to indicate which item types have been added to the knapsack. A *b-item type solution* solves the b-item type restricted kIKPSW. Note that a b-item type solution is not necessarily integer feasible. By Lemma 6.1, the 1-item type or 2-item type solution with the largest value solves the limited relaxation.

The kIKPSW Greedy heuristic, labeled $H_k^{1/2}$, yields solutions that are within a factor of 1/2 of the optimal value by using no more than two item types. First, $H_k^{1/2}$ finds the item type $j^* \in S^{k^+}$ such that $v_{j^*} \geq v_i$, $i \in S^{k^+}$. A trivial feasible solution is $k\, v_{j^*}$. $H_k^{1/2}$ then finds the 2-item type solution with the largest value by considering all possible pairs of item types. The resulting linear programming model can be solved in constant time. Note that the 2-item type solution with the largest value has one item type $j^+ \in S^{k^+}$ and one item type $j^- \in S^{k^-}$. Let the 2-item type solution contain x_{j^+} and x_{j^-} (possibly fractional) copies of item types j^+ and j^-, respectively. An integer feasible 2-item type solution to kIKPSW is produced by adding $\lceil x_{j^+} \rceil$ and $\lfloor x_{j^-} \rfloor$ copies of item types j^+ and j^- to the knapsack, respectively. This integer solution is feasible with respect to capacity, since $w_{j^-} > w_{j^+}$, or otherwise $x_{j^-} = k$. $H_k^{1/2}$ then returns the best 1-item or 2-item type integer feasible solution in $\mathrm{O}(n^2)$ time, where n^2 total pairs of item types are considered. Theorem 6.6 shows that $H_k^{1/2}$ yields a solution that is within a factor of 1/2 of the optimal solution.

The kIKPSW Greedy heuristic, labeled $H_k^{1/2}$, yields solutions that are within a factor of 1/2 of the optimal value by using no more than two item types. In the first step, $H_k^{1/2}$ finds the optimal solution using a single item type, $\max\{k\, v_i | k w_i + s_i \leq c\}$. $H_k^{1/2}$ then finds the optimal solution using exactly two item types by considering all possible pairs of item types. The resulting knapsack subproblems with two item types are formulated as linear programming models that can be solved in constant time. An integer feasible solution is created by rounding the fractional variables to create a feasible solution with respect to the capacity and the cardinality. $H_k^{1/2}$ then returns the best integer feasible solution with no more that two item types in $\mathrm{O}(n^2)$ time, where n^2 total pairs of item types are considered. Theorem 6.6 shows that $H_k^{1/2}$ yields a solution that is within a factor of 1/2 of the optimal solution.

Theorem 6.6 *$H_k^{1/2}$ is a 1/2-approximation algorithm.*

Proof. Without loss of generality, assume that that x_{j^+} and x_{j^-} are fractional. First, for the case

when $x_{j^-} \geq 1$, a feasible solution to kIKPSW can be produced by rounding down the value of x_{j^-} and rounding up the value of x_{j^+}. Then,

$$v_{j^-}\lfloor x_{j^-}\rfloor + v_{j^+}\lceil x_{j^+}\rceil = z^h \leq z \leq z^u = v_{j^-}x_{j^-} + v_{j^+}x_{j^+} \leq v_{j^-}(2\lfloor x_{j^-}\rfloor) + v_{j^+}\lceil x_{j^+}\rceil.$$

and, therefore, in the worst case, $(z - z^h)/z \leq 1/2$.

Now, for the case when $x_{j^-} < 1$, an upper bound is given by

$$z^u = v_{j^-}x_{j^-} + (k - x_{j^-})v_{j^+}$$

the corresponding integer feasible solution produced by $H_k^{1/2}$ is $z^h = kv_{j^+}$. The case $z^h < (1/2)z^u$ is of interest, yielding $k\, v_{j^+} < \frac{1}{2}(v_{j^-}x_{j^-} + v_{j^+}(k - x_{j^-}))$, which gives an upper bound

$$v_{j^+} < \frac{x_{j^-}}{k + x_{j^-}}v_{j^-}.$$

Given this bound for v_{j^+}, an upper bound for z^u is

$$z^u < v_{j^-}x_{j^-} + \left(\frac{x_{j^-}}{k + x_{j^-}}v_{j^-}\right)(k - x_{j^-}) < 2v_{j^-}.$$

By assumption, there is an integer feasible solution using at least one copy of item type j^-, which implies the existence of a solution using two item types. To see this, consider forming a feasible solution using exactly two item types from a solution using three or more item types with two of the three or more item types in the current solution, namely item types j^- and $t \in S^{k^+}$. Then, a (possibly non-integer) 2-item type solution is produced using $H_k^{1/2}$ with value $\bar{z}$, where $\bar{z} = v_{j^-}\bar{x} + v_t(k - \bar{x})$, with $\bar{x} \geq 1$, where $\bar{z} \leq z \leq z^u$. In the worst case, $v_t = 0$ and $\bar{x} = 1$, and hence $\bar{z} = v_{j^-}$ and

$$\bar{z} \leq z \leq z^u < 2\bar{z}.$$

Since $z^h \geq \bar{z}$, then $(z - z^h)/z \leq 1/2$. □

Example 2 shows that the bound of 1/2 given in Theorem 6.6 for $H^{1/2}$ is tight.

Example 2: Consider the series of examples with $n = 3$, $w_1 = v_2 = 1$, $s_1 = s_2 = v_1 = 0$, $w_2 = 2$, $s_3 = 3$, $w_3 = v_3 = k$, $c = 3k$. Item types 1 and 2 are in S^{k^+} and item type 3 is in S^{k^-}. The best integer feasible 1-item type solution value is k, using item type 2. The best integer feasible 2-item

type solution is k, with one copy of item type 3 and $k-1$ copies of item type 1. The optimal value is $2k-2$, with one copy each of types 1 and 3 and $k-2$ copies of item type 2. The bound is tight since $z^h/z = k/(2k-2)$ is arbitrarily close to $1/2$ for k sufficiently large. □

Corollary 6.2 provides a bound on the absolute error of the solution produced by $H_k^{1/2}$ and shows that half of the upper bound found by $H_k^{1/2}$ provides a lower bound for the

Corollary 6.2 *The following two results hold for* $H_k^{1/2}$*:*

(a) $H_k^{1/2}$ *yields a solution to kIKPSW that has absolute error no greater than* $(v_{j^-}) - (v_{j^+})$*, if* $H_k^{1/2}$ *uses item types* $j^+ \in S^{k^+}$ *and* $j^- \in S^{k^-}$*.*

(b) $z^u/2 \leq z$*.*

Proof. (a) The absolute error is bounded above by

$$v_{j^-}(x_{j^-} - \lfloor x_{j^-} \rfloor) - v_{j^+}(x_{j^+} - \lceil x_{j^+} \rceil) \leq v_{j^-} - v_{j^+}.$$

(b) Follows directly from $z^u \leq 2z^h$ □

The kIKPSW approximation algorithm, $F_k(\varepsilon)$, is a modification of $F(\varepsilon)$. In particular, the heuristic $H_k^{1/2}$ is run in the preprocessing phase of $F_k(\varepsilon)$ to first find a lower bound on the optimal solution value. In addition, let the threshold value be $\theta = \frac{1}{2}\varepsilon z^h$ and the scale factor be $\delta = \frac{1}{2}\varepsilon z^h/k$. $F_k(\varepsilon)$ obtains subsets of large and small item types to be used in the dynamic programming and Greedy phases, respectively, as used in $F(\varepsilon)$.

The second phase of $F_k(\varepsilon)$ solves kIKPSW over the large item types with scaled values with kDP-V, with the trivial modification to kDP-V to account for multiple copies of the large item types being added at a time. Note that kDP-V finds all solutions to kIKPSW that use no more than k items, since the remaining items may be added to the knapsack in the Greedy phase. Note that an upper bound is $O(1/\varepsilon^2)$ since the total number of scaled values considered in the dynamic programming phase is bounded above by $4k/\varepsilon$ and at most $4/\varepsilon$ large items are in a feasible solution. However, the largest cardinality possible is k since some large item types add multiple copies at a time. Hence, the dynamic programming phase requires $O(kn/\varepsilon^2)$ time and $O(k/\varepsilon^2)$ space.

During the Greedy phase, the third phase of $F_k(\varepsilon)$, the small items are inserted into each dynamic programming solution using $H_k^{1/2}$ with the remaining capacity and the remaining cardinality. For each of the $O(k/\varepsilon^2)$ solution sets in the dynamic programming phase, the Greedy phase requires $O(1/\varepsilon)$ to construct each solution set from the dynamic programming phase, and $O(n^2)$

time to apply $H_k^{1/2}$. Therefore, the Greedy phase requires $O(kn^2/\varepsilon^3)$ time. Hence, the overall time and space requirements of $F_k(\varepsilon)$ are $O(kn^2/\varepsilon^3)$ and $O(k/\varepsilon^2)$, respectively.

Theorem 6.7 shows that the solution found by $F_k(\varepsilon)$ is within a factor of ε of the optimal solution value for kIKPSW.

Theorem 6.7 *$F_k(\varepsilon)$ is an FPTAS for kIKPSW.*

Proof. The proof is identical to Theorem 6.5 except for the following changes. Although, $H_k^{1/2}$ is used instead of $H^{1/2}$, the absolute error made by $H_k^{1/2}$ is no greater than v_{max}. A dynamic programming solution contains more than k items, hence the overall error is less then ε. □

6.4 BSKP

This section addresses a generalization of BKP called the Bounded Set-up Knapsack Problem (BSKP). This section presents five dynamic programming algorithms that solve BSKP in pseudo-polynomial time using the characteristics of BSKP. In addition, a Greedy heuristic that obtains solutions whose values are at least half of the optimal solution value is described. Moreover, a FPTAS is presented that obtains solutions whose values are within an arbitrary level ε of the optimal solution value.

A key contribution of this section is that the algorithms and heuristics presented for BSKP also provide efficient methods for solving and approximating BKP. The most efficient way to solve BKP is to transform BKP into a KP instance and apply an algorithm designed for KP rather than apply a specialized algorithm for BKP [45] with several exceptions [44, 26, 63, 66]. The algorithms and heuristics presented in this section can be applied to BKP without any modifications since BKP is a particular case of BSKP. One of the dynamic programming algorithms for BSKP has the same time and space complexity for solving BKP as the best dynamic programming algorithm for BKP [63]. The FPTAS for BSKP also obtains approximate solutions to BKP with a better asymptotic time bound as compared to those obtained from using a FPTAS designed for KP to obtain approximate solutions to BKP.

This section is organized as follows. Section 6.4.1 describes four pseudo-polynomial time dynamic programming algorithms for solving BSKP. Section 6.4.2 presents two approximation algorithms for BSKP, including a Greedy heuristic and an FPTAS for BSKP, which obtain solutions to BSKP that are within a factor of 1/2 and $\varepsilon \in (0,\ 1/2]$ of the optimal solution value, respectively.

Section 6.4.3 analyzes the performance of the dynamic programming algorithms and the FPTAS in Sections 6.4.1 and 6.4.2 when applied to BKP.

6.4.1 Dynamic Programming Algorithms

This section introduces five dynamic programming algorithms for BSKP. The first three algorithms solve BSKP over the set of item types and the knapsack capacity. The next dynamic programming algorithm solves BSKP over the item types and objective function value. The last algorithm solves BSKP using lists. Four of the algorithms obtain the optimal solution set and its value, and one of the algorithms only obtains the optimal value.

The first dynamic programming algorithm (labeled DP-W1) uses a nested approach to solve BSKP in O(nc) time and space. To describe DP-W1, let $z_r(\bar{c})$, $r = 1, 2, \ldots, n$, $\bar{c} = 0, 1, \ldots, c$, be the optimal solution value to the following knapsack subproblem defined on the first r item types:

$$z_r(\bar{c}) = \max \left\{ \sum_{i=1}^{r} v_i x_i + \sum_{i=1}^{r} u_i y_i \left| \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r} s_i y_i \leq \bar{c};\ x_i \leq b_i, \right. \right. \tag{6.14}$$

$$\left. \left. i = 1, 2, \ldots, r;\ \frac{1}{b_i} x_i - y_i \leq 0,\ i = 1, 2, ..., r,\ y_i \leq x_i,\ i = 1, 2, \ldots, r \right\} \right.,$$

with decision variables $x_i \in Z_0^+$ and $y_i \in \{0, 1\},\ i = 1, 2, ..., n$. The corresponding optimal number of copies of item type r is $x_r(\bar{c})$, which ensures that $z_r(\bar{c})$ corresponds to a feasible solution with respect to item type r.

Let $\bar{z}_r(\bar{c})$ be the optimal value over the first $r = 1, 2, \ldots, n$ item types with knapsack capacity $\bar{c} = 0, 1, \ldots, c$ given such that $1 \leq \bar{x}_r(\bar{c}) < b_r$ copies of item type r are present in the knapsack. Let $\bar{z}_r^*(\bar{c})$ is the best objective function value over the first r item types with capacity $\bar{c}$, given that b_r copies of item type r are in the knapsack. Let $\bar{z}_r(\bar{c})$, $r = 1, 2, \ldots, n$, $\bar{c} = w_r + s_r, \ldots, c$, be an optimal solution to the following knapsack subproblem ($\bar{z}_r(\bar{c}) = \infty$ otherwise):

$$\bar{z}_r(\bar{c}) = \max \left\{ \sum_{i=1}^{r} v_i x_i + \sum_{i=1}^{r} u_i y_i \left| \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r-1} s_i y_i + s_r \leq \bar{c};\ \frac{1}{b_i} x_i - y_i \leq 0; \right. \right.$$
$$\left. \left. i = 1, 2, ..., r-1;\ x_i \leq b_i - 1,\ i = 1, 2, \ldots, r; y_i \leq x_i,\ i = 1, 2, \ldots, r;\quad x_r \geq 1 \right\} \right.,$$

with decision variables $x_i \in Z_0^+$ and $y_i \in \{0, 1\},\ i = 1, 2, ..., n$. Let $\bar{z}_r^*(\bar{c})$, $r = 1, 2, \ldots, n$, $\bar{c} =$

$w_r b_r + s_r, \ldots, c$ be an optimal solution to the following knapsack subproblem ($\bar{z}_r^*(\bar{c}) = \infty$ otherwise):

$$\bar{z}_r(\bar{c}) = \max \left\{ v_r b_r + u_r \sum_{i=1}^{r-1} v_i x_i + \sum_{i=1}^{r-1} u_i y_i \left| \sum_{i=1}^{r-1} w_i x_i + \sum_{i=1}^{r-1} s_i y_i + w_r b_r + s_r \leq \bar{c};\ \frac{1}{b_i} x_i - y_i \leq 0; \right. \right.$$
$$\left. \left. i = 1, 2, ..., r-1;\ y_i \leq x_i,\ i = 1, 2, \ldots, r-1 \right\},$$

Note that five values are stored for each pair of values $r = 1, 2, \ldots, n$, and $\bar{c} = 0, 1, \ldots, c$.

At each step of the recursion, $z_r(\bar{c})$ adds either no items of type r to the knapsack, adds at least one copy of item type r to the knapsack while $\bar{z}_r(\bar{c})$ adds at least one copy of item type r, or adds b_r copies of item type r to the knapsack. In this way, $z_r(\bar{c})$, $\bar{z}_r(\bar{c})$ and $\bar{z}_r^*(\bar{c})$ work together to identify the optimal solution. Note that $z_r(\bar{c})$ is feasible for item type $r = 1, 2, \ldots, n$, resulting in a feasible solution over all n item types. DP-W1 finds an optimal solution to BSKP in $O(nc)$ time since each step in the recursion calls at most two other recursions, and requires $O(nc)$ space. DP-W1 is described in pseudocode by Algorithm 5.

Algorithm DP-W1 finds the optimal solution set to BSKP and its value. In some applications, it may be necessary only to find the optimal value of BSKP. In this case, DP-W1 can be modified, reducing the space bound to $O(n + c)$ while keeping the time bound of $O(nc)$. The resulting algorithm, DP-W2 is described in pseudocode by Algorithm 6.

The optimal value and solution set of BSKP can be obtained in $O(nc)$ time and $O(n+c)$ space by applying a storage reduction scheme using a recursive "divide and conquer" approach [35, 63]. To solve BSKP, the set of items are divided such that the cardinality of each set is approximately equal, creating two subproblems. BSKP is solved over each subset of items given capacity $0, 1, \ldots, c$ using DP-W2. The solutions to both subproblems are combined, and the optimal number of items of one item type from each subproblem is known. Each subproblem is then divided into two more subproblems, and this process is repeated until the optimal solution set is known. Pferschy [63] shows how this approach requires $O(nc)$ time and $O(n + c)$ space. The resulting algorithm, DP-DC, is described in pseudocode by Algorithm 7. DP-DC takes a set of n' item types and knapsack capacity c', and $n' = n$ and $c' = c$ initially.

The fourth dynamic programming algorithm (labeled DP-V1) performs dynamic programming over the values rather than over the weights. DP-V1 uses a nested approach similar to that used in DP-W1 to solve BSKP in $O(nU)$ time and space, where U is an upper bound on the objective function value. To describe DP-V1, let $z_r(d)$ be a minimum weight solution value over the first

Algorithm 5 BSKP Dynamic Programming Algorithm DP-W1

set $z_r(\bar{c}) \leftarrow 0,\ x_r(\bar{c}) \leftarrow 0,\ r = 1, 2, \ldots, n,\ \bar{c} = 0, 1, \ldots, c$
set $\bar{z}_r(\bar{c}) \leftarrow \bar{z}_r^*(\bar{c}) \leftarrow -\infty,\ \bar{x}_r(\bar{c}) \leftarrow 0,\ r = 1, 2, \ldots, n,\ \bar{c} = 0, 1, \ldots, c$
for $\bar{c} \leftarrow w_1 + s_1$ to c **do**
  Comment: Set boundary conditions
  $x_1(\bar{c}) \leftarrow \min\{b_1,\ \lfloor(\bar{c} - s_1)/w_1\rfloor\},\ \bar{x}_1(\bar{c}) \leftarrow \min\{b_1 - 1,\ \lfloor(\bar{c} - s_1)/w_1\rfloor\},$
  $z_1(\bar{c}) \leftarrow v_1 x_1(\bar{c}) + u_1$
  if $w_1 b_1 + s_1 \leq \bar{c}$ **then**
    $\bar{z}_1^*(\bar{c}) \leftarrow u_1 + v_1 b_1$
  end if
  if $\bar{x}_1(\bar{c}) > 0$ **then**
    $\bar{z}_1(\bar{c}) \leftarrow v_1 \bar{x}_1(\bar{c}) + u_1$
  end if
end for
for $r \leftarrow 2$ to n **do**
  for $\bar{c} \leftarrow 0$ to $s_r + w_r - 1$ **do**
    $z_r(\bar{c}) \leftarrow z_{r-1}(\bar{c})$
  end for
  for $\bar{c} \leftarrow s_r + w_r$ to c **do**
    Comment: Update $\bar{z}_r^*(\bar{c})$
    if $w_r b_r + s_r \leq \bar{c}$ **then**
      $\bar{z}_r^*(\bar{c}) \leftarrow z_{r-1}(\bar{c} - s_r - w_r b_r) + u_r + v_r b_r$
    end if
    if $b_r > 1$ **then**
      Comment: Update $\bar{z}_r(\bar{c})$ *and* $\bar{x}_r(\bar{c})$
      if $(z_{r-1}(\bar{c} - s_r - w_r) + u_r + v_r \geq \bar{z}_r(\bar{c} - w_r) + v_r)$ OR $(\bar{x}_r(\bar{c} - w_r) = b_r - 1)$ **then**
        $\bar{z}_r(\bar{c}) \leftarrow z_{r-1}(\bar{c} - s_r - w_r) + u_r + v_r,\ \bar{x}_r(\bar{c}) \leftarrow 1$
      else
        $\bar{z}_r(\bar{c}) \leftarrow \bar{z}_r(\bar{c} - w_r) + v_r,\ \bar{x}_r(\bar{c}) \leftarrow \bar{x}_r(\bar{c} - w_r) + 1$
      end if
    else
      $\bar{z}_r(\bar{c}) \leftarrow -\infty,\ \bar{x}_r(\bar{c}) \leftarrow 0$
    end if
    if $z_{r-1}(\bar{c}) \geq \bar{z}_r(\bar{c})$ AND $z_{r-1}(\bar{c}) \geq \bar{z}_r^*(\bar{c})$ **then**
      Comment: Update $z_r(\bar{c})$ *and* $x_r(\bar{c})$
      $z_r(\bar{c}) \leftarrow z_{r-1}(\bar{c}),\ x_r(\bar{c}) \leftarrow 0$
    else if $z_{r-1}(\bar{c}) < \bar{z}_r(\bar{c})$ AND $\bar{z}_r(\bar{c}) \geq \bar{z}_r^*(\bar{c})$ **then**
      $z_r(\bar{c}) \leftarrow \bar{z}_r(\bar{c}),\ x_r(\bar{c}) \leftarrow \bar{x}_r(\bar{c})$
    else
      $z_r(\bar{c}) \leftarrow \bar{z}_r^*(\bar{c}),\ x_r(\bar{c}) \leftarrow b_r$
    end if
  end for
end for
Comment: Obtain X^{OPT}
$\bar{c} \leftarrow c$
for $r \leftarrow n$ down to 1 **do**
  $x_r^{OPT} \leftarrow x_r(\bar{c})$
  if $x_r^{OPT} > 0$ **then**
    $\bar{c} \leftarrow \bar{c} - s_r - w_r x_r^{OPT}$
  end if
end for
return $z^{OPT} \leftarrow z_n(c),\ X^{OPT}$

Algorithm 6 BSKP Dynamic Programming Algorithm DP-W2

set $z(\bar{c}) \leftarrow 0,\ x(\bar{c}) \leftarrow 0,\ \bar{c} = 0, 1, \ldots, c$
set $\bar{z}(\bar{c}) \leftarrow \bar{z}^*(\bar{c}) \leftarrow -\infty,\ \bar{x}(\bar{c}) \leftarrow 0,\ \bar{c} = 0, 1, \ldots, c$
for $\bar{c} \leftarrow w_1 + s_1$ to c **do**
  Comment: Set boundary conditions
  $x(\bar{c}) \leftarrow \min\{b_1,\ \lfloor(\bar{c} - s_1)/w_1\rfloor\},\ \bar{x}(\bar{c}) \leftarrow \min\{b_1 - 1,\ \lfloor(\bar{c} - s_1)/w_1\rfloor\},$
  $z(\bar{c}) \leftarrow v_1 x(\bar{c}) + u_1$
  if $w_1 b_1 + s_1 \leq \bar{c}$ **then**
    $\bar{z}^*(\bar{c}) \leftarrow u_1 + v_1 b_1$
  end if
  if $\bar{x}(\bar{c}) > 0$ **then**
    $\bar{z}(\bar{c}) \leftarrow v_1 \bar{x}(\bar{c}) + u_1$
  end if
end for
for $r \leftarrow 2$ to n **do**
  for $\bar{c} \leftarrow 0$ to $s_r + w_r - 1$ **do**
    $x(\bar{c}) \leftarrow 0,\ \bar{x}(\bar{c}) \leftarrow 0$
    $\bar{z}(\bar{c}) \leftarrow -\infty,\ \bar{z}^*(\bar{c}) \leftarrow -\infty$
  end for
  for $\bar{c} \leftarrow s_r + w_r$ to c **do**
    Comment: Update $\bar{z}_r^*(\bar{c})$
    $\bar{z}^*(\bar{c}) \leftarrow z(\bar{c} - s_r - w_r b_r) + u_r + v_r b_r$
  end for
  for $\bar{c} \leftarrow s_r + w_r$ to c **do**
    Comment: Update $\bar{z}_r(\bar{c})$ *and* $\bar{x}_r(\bar{c})$
    $x(\bar{c}) \leftarrow 0,\ \bar{x}(\bar{c}) \leftarrow 0$
    if $b_r > 1$ **then**
      if $z(\bar{c} - s_r - w_r) + u_r + v_r \geq \bar{z}(\bar{c} - w_r) + v_r$ OR $\bar{x}(\bar{c} - w_r) = b_r - 1$) **then**
        $\bar{z}(\bar{c}) \leftarrow z_{r-1}(\bar{c} - s_r - w_r) + u_r + v_r,\ \bar{x}(\bar{c}) \leftarrow 1$
      else
        $\bar{z}(\bar{c}) \leftarrow \bar{z}(\bar{c} - w_r) + v_r,\ \bar{x}(\bar{c}) \leftarrow \bar{x}(\bar{c} - w_r) + 1$
      end if
    else
      $\bar{z}(\bar{c}) \leftarrow -\infty,\ \bar{x}(\bar{c}) \leftarrow 0$
    end if
  end for
  for $\bar{c} \leftarrow s_r + w_r$ to c **do**
    Comment: Update $z_r(\bar{c})$ *and* $x_r(\bar{c})$
    if $z(\bar{c}) < \bar{z}(\bar{c})$ AND $\bar{z}(\bar{c}) \geq \bar{z}^*(\bar{c})$ **then**
      $z(\bar{c}) \leftarrow \bar{z}(\bar{c}),\ x(\bar{c}) \leftarrow \bar{x}(\bar{c})$
    else if $\bar{z}^*(\bar{c}) > z(\bar{c})$ AND $\bar{z}^*(\bar{c}) > \bar{z}(\bar{c})$ **then**
      $z(\bar{c}) \leftarrow \bar{z}^*(\bar{c}),\ x(\bar{c}) \leftarrow b_r$
    end if
  end for
end for
return $z^{OPT} \leftarrow z(\bar{c}),\ x(\bar{c}),\ \bar{c} = 0, 1, \ldots, c$

Algorithm 7 Dynamic Programming Algorithm DP-DC

Partition the set of n' items into subsets n_1 and n_2 with $|n_1| \approx |n_2|$.
Solve the first subproblem with n_1 item types and capacities $0, 1, \ldots, c'$, yielding $z_1(\bar{c})$ and $x_1(\bar{c})$, $\bar{c} = 0, 1, \ldots, c'$
Solve the second subproblem with n_2 item types and capacities $0, 1, \ldots, c'$, yielding $z_2(\bar{c})$ and $x_2(\bar{c})$, $\bar{c} = 0, 1, \ldots, c'$
Find $c_1,\ c_2$ such that $c_1 + c_2 = c'$ and $z_1(c_1) + z_2(c_2)$ is maximized
if $|n_1| \leq 1$ **then**
  Combine $x_1(c_1)$ with the optimal solution
else
  Use DP-DC to solve the BSKP instance of item types n_1 with capacity c_1
end if
if $|n_2| \leq 1$ **then**
  Combine $x_2(c_2)$ with the optimal solution
else
  Use DP-DC to solve the BSKP instance of item types n_2 with capacity c_2
end if
return $z^{OPT},\ X^{OPT}$

$r = 1, 2, \ldots, n$ item types with value $d = 0, 1, \ldots, U$. Define $z_r(d)$ as

$$z_r(d) = \min \left\{ \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r} s_i y_i \,\middle|\, \sum_{i=1}^{r} v_i x_i + \sum_{i=1}^{r} u_i y_i = d;\ x_i \leq b_i,\ i = 1, 2, ..., r, \right.$$

$$\left. i = 1, 2, \ldots, r;\ \frac{1}{b_i} x_i - y_i \leq 0;\ y_i \leq x_i,\ i = 1, 2, \ldots, r \right\},$$

with $x_r(d)$ copies of item type r. The value of $x_r(d)$ ensures that $z_r(d)$ corresponds to a feasible solution with respect to item type r. Let $\bar{z}_r(d)$ be a minimum weight solution value over the first $r = 1, 2, \ldots, n$ item types with value $d = 0, 1, \ldots, U$ given that $1 \leq \bar{x}_r(d) < b_r$ copies of item type r are present in the knapsack. Let $\bar{z}_r^*(d)$ be the minimum weight over the first r item types with value d, given that b_r copies of item type r are in the knapsack. Let $\bar{z}_r(d)$, $r = 1, 2, \ldots, n$, $d = u_r + v_r, \ldots, U$, be a minimum weight solution to the following knapsack subproblem ($\bar{z}_r(d) = \infty$ otherwise):

$$\bar{z}_r(d) = \min \left\{ s_r + \sum_{i=1}^{r} w_i x_i + \sum_{i=1}^{r-1} s_i y_i \,\middle|\, \sum_{i=1}^{r} v_i x_i + \sum_{i=1}^{r-1} u_i y_i + u_r \geq d;\ \frac{1}{b_i} x_i - y_i \leq 0; \right.$$
$$\left. i = 1, 2, ..., r-1;\ 1 \leq x_r \leq b_r - 1,\ y_i \leq x_i,\ i = 1, 2, \ldots, r-1 \right\},$$

with decision variables $x_i \in Z_0^+$ and $y_i \in \{0, 1\}$, $i = 1, 2, ..., n$. Let $\bar{z}_r^*(d)$, $r = 1, 2, \ldots, n$, $d = u_r + v_r b_r, \ldots, U$, be a minimum weight solution to the following knapsack subproblem ($\bar{z}_r(d) = \infty$

otherwise):

$$\bar{z}_r^*(d) = \min \left\{ s_r + w_r b_r + \sum_{i=1}^{r-1} w_i x_i + \sum_{i=1}^{r-1} s_i y_i \,\middle|\, \sum_{i=1}^{r} v_i x_i + \sum_{i=1}^{r-1} u_i y_i + u_r + v_r b_r \geq d;\ \frac{1}{b_i} x_i - y_i \leq 0; \right.$$
$$\left. i = 1, 2, ..., r-1;\ y_i \leq x_i,\ i = 1, 2, \ldots, r-1 \right\},$$

with decision variables $x_i \in Z_0^+$ and $y_i \in \{0, 1\},\ i = 1, 2, ..., n$.

Note that five values are stored for each pair of values $r = 1, 2, \ldots, n$, and $d = 0, 1, \ldots, U$. The optimal solution of BSKP has objective function value $\max \{d \,|z_n(d) \leq c\}$. DP-V1 is described in pseudocode by Algorithm 8.

DP-V1 finds an optimal solution to BSKP in $O(nU)$ time and $O(nU)$ space. Two more dynamic programming algorithms (DP-V2 and DP-V3) can be designed to solve BSKP over the values by modifying DP-V1 in a similar manner as DP-W2 and DP-W3 modified DP-W1. DP-V2 obtains an optimal solution value to BSKP in $O(nU)$ time and $O(n + U)$ space, and DP-V3 produces an optimal solution and its value to BSKP in $O(n^2U)$ time and $O(n + U)$ space.

The final dynamic programming algorithm solves BSKP with lists. List L_r contains a series of m entries over the first r item types

$$L_r = \langle (V_1, W_1, I_1), (V_2, W_2, I_2), \ldots, (V_m, W_m, I_m) \rangle,$$

where V_r denotes the value (including set-up values) of the partial knapsack solution, W_r denotes the corresponding weight (including set-up weights) of the partial knapsack solution, and I_r is the set of items in the knapsack and their multiplicity.

Consider two entries in a list, $(V_1, W_1, I_1),\ (V_2, W_2, I_2)$. The first state *dominates* the second if $W_1 < W_2$ and $V_1 \geq V_2$ or if $W_1 \leq W_2$ and $V_1 > V_2$. Dominated solutions never lead to an optimal solution of BSKP [37] and can be removed from the list. Moreover, states with weights greater than the knapsack capacity or values greater than the upper bound can be removed from the list. Assume that the dominated entries are removed when lists are merged, and that the lists are sorted in increasing value and weight. The advantage of solving BSKP with lists is that the storage requirements are low. Since most of the states stored by algorithms DP-W1 and DP-V1 are dominated by other solutions, they do not need to be stored. This motivates dynamic programming algorithm DP-L1, described in pseudocode by Algorithm 9. In order to solve BSKP, items must be added to partial knapsack solutions in order of increasing weight and value.

Algorithm 8 BSKP Dynamic Programming Algorithm DP-V1

$z_r(0) \leftarrow 0,\ x_r(0) \leftarrow 0,\ \bar{z}_r(0) \leftarrow \bar{z}_r^*(0) \leftarrow \infty,\ \bar{x}_r(0) \leftarrow 0$
for $d \leftarrow 1$ to U **do**
  $x_1(d) \leftarrow \bar{x}_1(d) \leftarrow \max\{1, \lceil (d-u_1)/v_1 \rceil\}$
  $z_1(d) \leftarrow \bar{z}_1(d) \leftarrow s_1 + w_1 x_1(d),\ \bar{z}_1^*(d) \leftarrow \infty$
  if $x_1(d) > b_1$ $(\bar{x}_1(d) > b_1 - 1)$ **then**
    $z_1(d) \leftarrow \infty,\ x_1(d) \leftarrow 0$ $(\bar{z}_1(d) \leftarrow \infty,\ \bar{x}_1(d) \leftarrow 0)$
  end if
  if $d \leq u_1 + v_1 b_1$ **then**
    $\bar{z}_1^*(d) \leftarrow s_1 + w_1 b_1$
  end if
end for
for $r \leftarrow 2$ to n **do**
  for $d \leftarrow 1$ to U **do**
    if $d \leq u_1 + v_1 b_1$ **then**
      $\bar{z}_r^*(d) \leftarrow s_1 + w_1 b_1$
    else if $z_{r-1}(d - u_r - v_r b_r) \neq \infty$ **then**
      $\bar{z}_r^*(d) \leftarrow s_1 + w_1 b_1 + z_{r-1}(d - u_r - v_r b_r)$
    end if
    $\bar{d}_0 \leftarrow \max\{0,\ d - v_r\},\ \bar{d}_1 \leftarrow \max\{0,\ d - v_r - u_r\}$
    if $b_r > 1$ **then**
      Comment: Update $\bar{z}_r(d)$ *and* $\bar{x}_r(d)$
      if $z_{r-1}(\bar{d}_1) + s_r + w_r \leq \bar{z}_r(\bar{d}_0) + w_r$ OR $\bar{x}_r(\bar{d}_0) = b_r - 1$ **then**
        $\bar{z}_r(d) \leftarrow z_{r-1}(\bar{d}_1) + s_r + w_r,\ \bar{x}_r(d) \leftarrow 1$
        if $\bar{x}_r(\bar{d}_0) = b_r - 1$ **then**
          $\bar{z}_r^*(d) \leftarrow \min\{\bar{z}_r^*(d),\ \bar{z}_r(\bar{d}_0) + w_r\}$
        end if
      else
        $\bar{z}_r(d) \leftarrow \bar{z}_r(\bar{d}_0) + w_r,\ \bar{x}_r(d) \leftarrow \bar{x}_r(\bar{d}_0) + 1$
      end if
    else
      $\bar{z}_r(d) \leftarrow \infty,\ \bar{x}_r(d) \leftarrow 0$
    end if
    if $z_{r-1}(d) \leq \bar{z}_r(d)$ AND $z_{r-1}(d) \leq \bar{z}_r^*(d)$ **then**
      Comment: Update $z_r(d)$ *and* $x_r(d)$
      $z_r(d) \leftarrow z_{r-1}(d),\ x_r(d) \leftarrow 0$
    else if $\bar{z}_r(d) < z_{r-1}(d)$ AND $\bar{z}_r(d) \leq \bar{z}_r^*(d)$ **then**
      $z_r(d) \leftarrow \bar{z}_r(d),\ x_r(d) \leftarrow \bar{x}_r(d)$
    else
      $z_r(d) \leftarrow \bar{z}_r^*(d),\ x_r(d) \leftarrow b_r$
    end if
  end for
end for
$z^{OPT} \leftarrow \max\{d | z_n(d) \leq c\},\ d \leftarrow z^{OPT}$
for $r \leftarrow n$ down to 1 **do**
  $x_r^{OPT} \leftarrow x_r(d)$
  if $x_r^{OPT} > 0$ **then**
    $d \leftarrow d - u_r - v_r x_r^{OPT}$
  end if
end for
return $z^{OPT},\ X^{OPT}$

Algorithm 9 BSKP Dynamic Programming Algorithm DP-L1

$L_0 = \langle (0, 0, \emptyset) \rangle$
for $r = 1$ to n **do**
  Add item type r (including set-up weights and values) to all elements in L_{r-1}, yielding L'_{r-1}.
  Add item type r (*not* including set-up weights and values) to all elements in L'_{r-1}, keeping between 1 and $b_r - 1$ copies of item type r, yielding L'_{r-1}.
  Add b_r copies of item type r (including set-up weights and values) to all elements in L_{r-1}, yielding L''_{r-1}.
  L_r merges L_{r-1}, L'_{r-1}, and L''_{r-1}.
end for
return the largest state in L_n

The items are added to existing entries in the list from smallest to largest weight and value. Note that first item type can be added to all items in the current list as long as there is room in the knapsack. Additional items of type $r = 1, 2, \ldots, n$ can be added, but the bound b_r must be first checked. The two lists can be merged in $O(\min\{c, U\})$ time using pointers, and their method is trivially modified to merge three lists [37]. The resulting algorithm requires $O(n \min\{c, U\})$ time and $O(n \min\{c, U\})$ space.

6.4.2 Heuristics

This section presents two heuristics for BSKP. The BSKP Greedy heuristic, labeled $H_B^{1/2}$, obtains solutions that are within a factor of 1/2 of the optimal solution in $O(n)$ time by constructing an integer feasible solution from the linear programming relaxation of BSKP. To describe $H_B^{1/2}$, define a *reward* as an item type $i \in R$ with bound $b_i > 1$ such that either $u_i/s_i > v_i/w_i$ if $s_i > 0$, or $u_i > 0$ if $s_i = 0$. Alternatively, define a *penalty* item type $i \in P$ as an item type such that either $u_i/s_i \leq v_i/w_i$ if $s_i > 0$, or $u_i = 0$ if $s_i = 0$. Note that R and P partition the set of item types, $\{1, 2, \ldots, n\}$.

Consider the continuous linear programming relaxation of BSKP, given by objective (6.6) and constraints (6.7), (6.8), (6.9), (6.15), and (6.16):

$$0 \leq x_i \leq b_i, \ i = 1, 2, \ldots, n, \tag{6.15}$$

$$0 \leq y_i \leq 1, \ i = 1, 2, \ldots, n. \tag{6.16}$$

The values for $y_i, \ i = 1, 2, \ldots, n$ are set by the set of constraints, (6.8) and (6.9). If $i \in P$, then $y_i = x_i/b_r$, which adds the set-up weight s_i and set-up value u_i over all b_i copies of item type i. If $i \in R$, then $y_i = x_i$ if $x_i < 1$ and $y_i = 1$ otherwise, which adds the set-up weight s_i and set-up

value u_i over the first copy of item type i added to the knapsack. Therefore, (6.8) and (6.9) can be rewritten for penalty and reward item types as

$$\frac{1}{b_i}x_i = y_i,\ i \in P \tag{6.17}$$

$$y_i = \min\{1,\ x_i\},\ i \in R. \tag{6.18}$$

Using constraints (6.17) and (6.18), the continuous linear programming relaxation of BSKP formulates an instance of the continuous linear programming relaxation of BKP, with $n' = |P|+2|R|$ item types, where each penalty item type $i \in P$ in BSKP is defined for BKP with value $v'_i = v_i + u_i/b_i$, weight $w'_i = w_i + s_i/b_i$, and bound $b'_i = b_i$. Each reward item type $i \in R$ can be broken into two item types with the first item type corresponding to the first copy and the second item type corresponding to the remaining $b_i - 1$ copies. Therefore, an item in i_1 in BKP is created with value $v'_{i_1} = v_i + u_i$, weight $w'_{i_1} = w_i + s_i$, and bound $b'_{i_1} = 1$. In addition, an item in i_0 in BKP is created with value $v'_{i_0} = v_i$, weight $w'_{i_0} = w_i$, and bound $b'_{i_0} = b_i - 1$. The decision variables are $x'_1, x'_2, \ldots, x'_{n'}$ with $0 \le x'_i \le b'_i,\ i = 1, 2, \ldots, n'$.

To illustrate $H_B^{1/2}$, assume that the item types in BKP are sorted such that

$$\frac{v'_1}{w'_1} \ge \frac{v'_2}{w'_2} \ge \cdots \ge \frac{v'_{n'}}{w'_{n'}}. \tag{6.19}$$

Note that for a BSKP reward item i, the BKP item i_1 corresponding to the first copy occurs earlier in the sequence (as defined in (6.19)) than the BKP item i_0 corresponding to the remaining $b_i - 1$ copies. To see this, first consider the case when $s_i > 0$. Then, $u_i/s_i > v_i/w_i$, which can be arranged, and, after adding $w_i v_i$ to both sides, yields $w_i(u_i + v_i) > v_i(s_i + w_i)$. Therefore,

$$\frac{v'_{i_1}}{w'_{i_1}} = \frac{u_i + v_i}{s_i + w_i} > \frac{v_i}{w_i} = \frac{v'_{i_0}}{w'_{i_0}}.$$

Now consider the case when $s_i = 0$ and $u_i > 0$. Then,

$$\frac{v'_{i_1}}{w'_{i_1}} = \frac{u_i + v_i}{w_i} > \frac{v_i}{w_i} = \frac{v'_{i_0}}{w'_{i_0}}.$$

Let the critical item type a for the BKP instance be defined as

$$a = \min \left\{ j : \sum_{i=1}^{j} b'_i w'_i \right\}.$$

The optimal value of the continuous linear programming relaxation of the BKP instance is given by

$$x'_i = b'_i, \ i = 1, 2, \ldots, a-1,$$

$$x'_i = 0, \ i = a+1, a+2, \ldots, n',$$

$$x'_a = \frac{\bar{c}}{w'_a},$$

where $\bar{c} = c - \sum_{i=1}^{a-1} b'_i w'_i$. The continuous linear programming relaxation of BSKP is defined as follows. The number of copies of item type i in BSKP is the same as the number of copies of this item type in BKP. Note that this involves adding two values if $i \in R$. The values of $y_1, y_2, \ldots, y_n$ are determined by (6.17) and (6.18). Note that the critical item type can be found in linear time without sorting the item types in the BKP instance [37].

An integer feasible solution for BSKP can be constructed from the continuous linear programming relaxation of BSKP as follows. First note that, except for the critical item type, the number of copies of a penalty item type added to the knapsack is either zero or equal to the bound. Therefore, the same number of copies of the penalty items are added to the knapsack in the BSKP instance. Note that, except for the critical item type, the number of copies of a reward item type added to the knapsack is either zero, one, or equal to the bound. If only one copy is added, then the set-up value and set-up weight are included in the knapsack. Therefore, the same number of copies of the reward items is added to the knapsack in the BSKP instance. As many copies of the critical item type are added to the knapsack such that the remaining capacity $\bar{c}$ is not exceeded.

Let z^u denote the objective function value for the continuous linear programming relaxation for the BSKP instance, which is an upper bound on the objective function value for BSKP. Let z^g denote the objective function value to the integer feasible solution formed from the continuous linear programming relaxation. Let $z^b = \max_i\{u_i + b_i v_i\}$. Then, $H_B^{1/2}$ yields the objective function value

$$z^h = \max\{z^g, \ z^b\}.$$

The absolute error, given by $E = z - z^h$, where z is the optimal objective function value,

can be determined by the critical item type. Note that if the critical item type is a penalty item $i \in P$, then the absolute error is bounded by $v'_a x'_a = (v_i + u_i/b_i)x'_a < v_i b_i + u_i$. If the critical item type corresponds to the first copy of a reward item $i \in R$, then the absolute error is bounded by $v'_a x'_a < v_i + u_i$ since $x'_a < 1$. If the critical item type corresponds to the remaining copies of a reward item $i \in R$, then the absolute error is bounded by $v'_a(x'_a - \lfloor x'_a \rfloor) < v_i$ since $\lfloor x'_a \rfloor$ copies of item type i can be added to the knapsack to form an integer feasible solution. Therefore, the absolute error, E, is bounded by

$$z - z^h = E < \max\left\{\max\{v_i b_i + u_i \ : \ i \in P\},\ \max\{v_i + u_i \ : \ i \in R\}\right\}. \tag{6.20}$$

Theorem 6.8 shows that $H_B^{1/2}$ provides an integer feasible solution whose objective function value is greater than half of the optimal objective function value.

Theorem 6.8 *$H_B^{1/2}$ is a 1/2-approximation algorithm for BSKP.*

Proof. Let z represent the optimal solution value. Then, $z - z^h = E$, and

$z \leq z^h + E < z^h + z^b \leq 2z^h$. □

Example 1 shows that this 1/2 bound is tight.

Example 1: Consider the series of examples with $m = 3$, weights $w_1 = w_2 = w_3 = r$, set-up weights $s_1 = 1$, $s_2 = s_3 = 0$, values $v_1 = r + 1$ and $v_2 = v_3 = r$, set-up values $u_1 = u_2 = u_3 = 0$, and bounds $b_1 = b_2 = b_3 = 1$, and capacity $c = 2r$. Note that item types 2 and 3 are identical. The heuristic solution value is $r + 1$ since it adds one copy of item type 1 to the knapsack, and the optimal solution value is $2r$ since it adds one copy each of item types 2 and 3 to the knapsack. The bound is tight since $z^h/z = (r + 1)/2r$ is arbitrarily close to 1/2 for r sufficiently large. □

Corollary 6.3 shows that half the upper bound found by $H_B^{1/2}$ provides a lower bound for the optimal solution of BSKP.

Corollary 6.3 *Half of the upper bound obtained by $H_B^{1/2}$ provides a lower bound for the optimal solution, and hence, $z^u/2 \leq z$.*

Proof. Note that $z^u \leq z^g + E$. Since $z^g \leq z^h$ and $E \leq z^h$, then $z^u \leq 2z^h$. Therefore, $z^u/2 \leq z^h \leq z$. □

The BSKP approximation algorithm, $F_B(\varepsilon)$, obtains solutions whose values are within a factor of ε of the optimal solution value. There is a preprocessing phase and two main phases, the dynamic

programming and Greedy phases, in the execution of $F_B(\varepsilon)$ for $\varepsilon < 1/2$.

In the preprocessing phase, $H_B^{1/2}$ is called to find lower and upper bounds on the objective value, z^h and z^u, respectively. Let the threshold value be defined as $\theta = \varepsilon z^h/2$, and let the scale factor be defined as $\delta = \varepsilon^2 z^h/4$. The set of item types can be partitioned into four subsets, Dynamic Programming item types (D), Greedy item types (G), Transition Penalty item types (T_P) and Transition Reward item types (T_R). An item type $i \in P$ is a Dynamic Programming item if $v_i \geq \theta$, and it is a Greedy item type if $v_i b_i + u_i \leq \theta$. Otherwise, item type i is a Transition Penalty item type (i.e., $v_i b_i + u_i > \theta$ and $v_i < \theta$). An item type $i \in R$ is a Dynamic Programming item if $v_i \geq \theta$, and it is a Greedy item type if $v_i + u_i \leq \theta$. Otherwise, item type i is a Transition Reward item type (i.e., $v_i + u_i > \theta$ and $v_i < \theta$).

The Dynamic Programming, Transition Penalty and Transition Reward item types are used to form an instance of the *augmented problem* with $n_A = |D| + |T_R| + |T_P| \leq n$ item types as follows. An item type in the augmented problem is created for each item type $i \in D$ with weight w_i, set-up weight s_i, value v_i, set-up value u_i, and bound b_i. The items are added during the dynamic programming phase with scaled values $\lfloor (v_i + u_i)/\delta \rfloor$ for the first copy and $\lfloor v_i/\delta \rfloor$ for the remaining copies. An item type in the augmented problem is created for each item type $i \in T_R$ with weight w_i, set-up weight s_i, value v_i, set-up value u_i, and bound 1. The item is added during the dynamic programming phase with scaled value $\lfloor (v_i + u_i)/\delta \rfloor$. For every item type $i \in T_P$, first find the smallest integer q_i such that $q_i v_i > \theta$. An item type in the augmented problem is created for each item type $i \in T_P$ with weight $q_i w_i$, set-up weight s_i, value $q_i v_i$, set-up value u_i, and bound $\lfloor b_i/q_i \rfloor$. The items are added during the dynamic programming phase with scaled values $\lfloor (q_i v_i + u_i)/\delta \rfloor$ for the first item type and $\lfloor q_i v_i/\delta \rfloor$ for the remaining item types, adding q_i copies each time.

The dynamic programming algorithm DP-L1 solves the augmented problem using the scaled values instead of the values and set-up values. Note that DP-L1 considers at most

$$\left\lfloor \frac{z}{\delta} \right\rfloor \leq \frac{2z^h}{\frac{1}{4}\varepsilon^2 z^h} = \frac{8}{\varepsilon^2}$$

total states.

The Greedy phase uses $H_B^{1/2}$ to greedily insert the Greedy, Transition Penalty, and Transition Reward item types into the dynamic programming solutions. Copies of Greedy item types can be added to any dynamic programming solution. However, a Transition Penalty or a Transition Reward item type can only be added to dynamic programming solution with at least one copy of

that item type in the knapsack, with the set-up weight added to the knapsack and the set-up value included in the objective function value. Therefore, the set-up weight and the set-up value for any item type $i \in T_P$ or $i \in T_R$ are set to zero in the Greedy phase.

The Greedy phase initially creates a BKP instance corresponding to the continuous programming relaxation of the Greedy, Transition Penalty, and Transition Reward item types as in $H_B^{1/2}$. The item types considered in the Greedy phase are inserted into each defined state considered in the dynamic programming phase. A Transition Reward or a Transition Penalty item type are not considered if they are not present in the dynamic programming solution. Therefore, the Greedy phase requires $O(n/\varepsilon^2)$ time to execute.

The total time and space bounds for $F_B(\varepsilon)$ are $O(n/\varepsilon^2)$ and $O(n + 1/\varepsilon^3)$, respectively. To see this, note that the Greedy phase can be executed in $O(n/\varepsilon^2)$ time. The dynamic programming phase requires $O(n/\varepsilon^2)$ time since the dynamic programming phase considers at most $8/\varepsilon^2$ states. Since there are no more than $4/\varepsilon$ items in the dynamic programming solution, the dynamic programming phase requires $O(1/\varepsilon^3)$ space. These bounds may be improved by using other techniques [41, 37].

Theorem 6.9 shows that the solution obtained by $F_B(\varepsilon)$ is within a factor of ε of the optimal solution.

Theorem 6.9 *$F_B(\varepsilon)$ is a FPTAS for BSKP.*

Proof. Let z^f be the value of the solution obtained by $F_B(\varepsilon)$, and let z be the optimal solution value. Error between the heuristic and optimal solutions accumulates in the dynamic programming and the Greedy phases. Note that for any item i in the augmented problem with value $\hat{v}_i$, set-up value $\hat{u}_i$, scaled value for the first copy added to the knapsack $\bar{u}_i$, and scaled value for the remaining copies $\bar{v}_i$,

$$\delta\bar{u}_i \leq \hat{v}_i + \hat{u}_i < \delta(\bar{u}_i + 1)$$

if the first copy of item type i is added, and

$$\delta\bar{v}_i \leq \hat{v}_i < \delta(\bar{v}_i + 1)$$

otherwise. Since the augmented problem is defined such that no more than z/θ items can fit in any of the dynamic programming solutions, then the error during the dynamic programming phase is limited to $\delta z/\theta$.

The absolute error made by $H_B^{1/2}$ in the Greedy phase is limited by θ. To see this, note that the absolute error is limited by

$$\max\{\max_{i\in G}\{v_i b_i + u_i\},\ \max_{i\in T_P}\{v_i\},\ \max_{i\in T_R}\{v_i\}\} \leq \max\{\theta,\ \theta,\ \theta\} = \theta$$

Therefore, the relative error is bounded above by ε since

$$\frac{z - z^f}{z} \leq \frac{\delta}{\theta} + \frac{\theta}{z} = \frac{\varepsilon^2 z^h/4}{\varepsilon z^h/2} + \frac{\varepsilon z^h/2}{z} \leq \varepsilon.$$

□

6.4.3 Application to the Bounded Knapsack Problem

There is a dearth of specialized algorithms and heuristics for BKP. The dynamic programming algorithms in Section 3 and the FPTAS $F_B^{1/2}$ can efficiently solve or approximate BKP.

The dynamic programming algorithms can be applied to solve BKP without modification. Therefore, DP-W1 solves BKP in $O(nc)$ time and space, DP-DC solves BKP in $O(nc)$ time and $O(n+c)$ space, and DP-L1 solves BKP in $O(n \min\{c, U\})$ time and space. DP-W2 finds the optimal value of BKP in $O(nc)$ time and $O(n + c)$ space.

Until recently, the best dynamic programming algorithm for BKP had time bound $O(nc^2)$ and space bound $O(nc)$ [45]. Pferschy [63] presents a dynamic programming algorithm that solves BKP in $O(nc)$ time and $O(n + c)$ space. Therefore, DP-DC solves BKP and BSKP, its generalization, with the same time and space requirements. It is unclear how the dynamic programming algorithm given by [63] could be modified to solve BSKP.

There are few FPTASs designed solely for BKP. FPTASs for KP typically are used to obtain approximate solutions to BKP by converting BKP to a KP instance [24, 41, 45]. Doing so creates an instance of KP with

$$\hat{n} = \sum_{i=1}^{n} \lceil \log_2(b_i + 1) \rceil$$

items. Once a KP instance is obtained, a number of FPTASs can be used to obtain approximate solutions for BKP. The FPTAS for KP in Lawler [41] requires $O(\hat{n} \log(1/\varepsilon) + 1/\varepsilon^4)$ time and $O(\hat{n} + 1/\varepsilon^3)$ space for BKP. The FPTAS for KP in Magazine and Oguz [43] requires $O((\hat{n})^2 \log \hat{n}/\varepsilon)$ time and $O(\hat{n}/\varepsilon)$ space for BKP. The FPTAS for KP in [35] requires $O(\hat{n} \log(1/\varepsilon) + (1/\varepsilon^4) \log(1/\varepsilon))$ time and $O(\hat{n} + 1/\varepsilon^2)$ space for BKP. This time bound is improved upon by Kellerer and Pferschy

[36] to $O(n \log(1/\varepsilon) + (1/\varepsilon^3)(\log(1/\varepsilon))^2)$. The bounds are evaluated such that they are asymptotic in n rather than in $1/\varepsilon$.

The FPTAS $F_B(\varepsilon)$ can be modified to obtain ε-approximate solutions to BKP in $O(n + 1/\varepsilon^3)$ time and $O(n + 1/\varepsilon^3)$ space. To see this, first note that the Greedy heuristic for BKP can be used instead of $H_B^{1/2}$. As noted previously, DP-L1 can be used for BKP without modification. Using a median-finding routine during the Greedy phase improves the time bound during this phase to $O(n + 1/\varepsilon^2)$ with no additional time requirements [41]. Therefore, $F_B(\varepsilon)$ has a better time complexity than applying any of the FPTASs designed for KP to BKP, although a better space bound is provided by [35, 36].

6.5 Conclusions

This chapter introduces and analyzes three knapsack variations, IKPSW, kIKPSW, and BSKP. These these problems are shown to be NP-hard. Several dynamic programming algorithms are designed to solve these problems in pseudo-polynomial time, including two for IKPSW, two for kIKPSW, and five for BSKP. Furthermore, for each problem, a Greedy heuristic that produces approximate solutions that are within a factor of 1/2 of the optimal solution and a FPTAS are presented. $H_k^{1/2}$, the Greedy heuristic for kIKPSW, uses no more than two item types, which suggests that near-optimal solutions to kIKPSW use few item types.

An implication of this research is that the dynamic programming algorithms and the FPTAS $F_B(\varepsilon)$ can be applied to BKP. DP-DC solves BKP in $O(nc)$ time and $O(n + c)$ space in the worst-case. These bounds match those of the best-known dynamic programming algorithm for BKP. $F_B(\varepsilon)$ obtains approximate solutions to BKP in $O(n + 1/\varepsilon^3)$ time and $O(n + 1/\varepsilon^2)$ space, which improves the asymptotic time bound for obtaining an ε-approximate solutions to BKP as compared to applying FPTASs designed for KP to BKP.

Chapter 7

Summary

This dissertation shows that operations research has the potential to shape the future of aviation security systems by summarizing several research efforts that apply operations research to passenger screening problems. However, there are some limitations. When doing operations research modeling (or in fact, mathematical modeling of any type), one must often make assumptions that may limit the applicability of the results obtained. Though such assumptions are often based on reasonable and realistic factors, they may pose difficulties in facilitating the transfer of the operations research analysis to decision-makers, since errors can lead to security breakdowns that may place people at an unnecessary risk. Second, some of the values used in the analysis are security-sensitive information and cannot be disseminated in the public domain. Instead, values that are representative of the real-world values were considered. Since real-world data was not always used, the conclusions drawn from the examples are limited. However, using more realistic data leads to more realistic conclusions. Third, the models presented measure the average or mean performance. In aviation security systems, average performance does not always capture the most interesting and salient aspects of such operations, which are often concerned with rare events and extreme events.

Determining how to deploy new screening technologies once security procedures are in place is challenging since it often takes a considerable amount of time to deploy and install new security devices. Before a device is fully deployed to all airports, there is a period of time when there is not sufficient capacity available to screen all passengers. The aviation security system can be viewed as a selective screening system when deploying new security devices under a uniform screening system. Therefore, the issues considered in this chapter will continue to remain important in the future, regardless of whether the TSA pursues uniform or selective screening strategies.

7.1 Extensions

The research presented in this dissertation focuses on designing multilevel passenger screening systems. The models introduced are one approach for designing effective passenger screening systems, and there are several possible directions to extend the research presented.

MAP and MPSP can be extended in the following ways.

- One possible extension is to consider modeling terrorists "gaming" the system. Since MAP and MPSP assume that the passengers' assessed threat values are an accurate representation of passenger risk, gaming the system is not applicable to the models as they are defined. One way to counter the gaming of the system is to treat the assessed threat values as random variables rather than as deterministic values. Iteratively sampling assessed threat values from a given density function and solving the model for each such sample problem yields a series of solutions that have different passenger assignments and that use different classes. In order for the system to be gamed, terrorists would then require a stochastic strategy since the passenger assignments change based on the assessed threat value samples.

- Minimizing the conditional false alarm rate is of interest because the majority of passengers are not threats, and resolving false alarms is expensive for the airlines and the Department of Homeland Security. This suggests the development of the Multicriteria MAP and the Multicriteria MPSP, which simultaneously minimize the conditional false clear rate and minimize the conditional false alarm rate. Alternatively, the objective could be redefined to minimizing the conditional false alarm rate, subject to a conditional false clear constraint and the remaining constraints of MAP or MPSP.

- The Sequential Stochastic MAP, an extension of MAP in which passengers arrive sequentially, can be formulated in a similar manner as SSMPSP. Studying algorithms that obtain optimal and near-optimal passenger assignment policies can shed light on real-time passenger assignment policies.

SSMPSP can be extended in the following ways.

- SSMPSP does not consider screening costs (i.e., the purchase and installation costs of screening devices and the direct costs of screening passengers with the device types), as it is currently defined. However, how screening devices are used affect the costs associated with passenger

screening. Extending SSMPSP to incorporate screening costs can give insight into the design of cost-effective screening strategies.

- SSMPSP can be extended to consider two objectives, minimizing the conditional false clear and false alarm rates. Designing policies that minimize both of these criteria provides insight into the design of robust, real-time passenger screening policies.

- Understanding how real-time models such as SSMPSP can be manipulated by terrorists wishing to be screened by less secure classes illustrates the weaknesses of such models. In particular, modeling how terrorists could game the system given how SSA works as well as manipulating the passenger pool (e.g., flooding the airport with decoys) may provide the impetus for designing robust heuristics and algorithms.

- SSMPSP models a multilevel screening system under which a large number of classes is potentially available. Analyzing a restricted version of SSMPSP that considers two classes can provide insight into the operation and limitation of the passenger screening system currently in operation at the nation's airports.

The analysis presented in Chapter 5 can be extended in the following ways.

- The analysis for checked baggage screening can be applied to the screening of passengers and their carry-on baggage. In this case, the screening costs incurred are expected to be dominated by labor costs, rather than the costs to purchase and install security devices. The analysis can also be extended to handle the screening of cargo.

- The analysis in Chapter 5 attempts to answer the following question in a systematic way: If there is an extra dollar available for checked baggage screening systems, should it be invested in developing more accurate intelligence or more accurate technology? Characterizing the tradeoffs between intelligence and technology is a difficult problem. Exploring alternative ways to model this tradeoff has the potential to influence how billions of dollars can be spent toward aviation security.

The knapsack model variations introduced in Chapter 6 can be extended in the following ways.

- Dynamic programming algorithms are presented for solving IKPSW, kIKPSW, and BSKP to optimality. Although these dynamic programming algorithms execute in pseudo-polynomial

time, they are not practical for solving large problem instances. One extension is to design more efficient methods to solve IKPSW and kIKPSW that exploit the characteristics of these problems, including the range of the weights and the set-up weight values, as well as the ratios of the values to weights.

- In most FPTASs for KP and IKP, the set of item types are partitioned into two groups (e.g., small and large), where the item types in the first group are used in the dynamic programming phase and the item types in the second group are used in the Greedy phase. In the FPTASs described in Chapter 6, the set of item types are not partitioned; rather, some item types may be used on both the dynamic programming and Greedy phases. This suggests exploring the design of new algorithms for knapsack problems, which consider new paradigms for implementing FPTASs.

The models and algorithms summarized in this dissertation can be applied to other security problems and problems in other domains. The following list contains several suggestions.

- Aviation security is composed of many components. Protecting aircraft, as considered by this dissertation, is one of these components. The models in this dissertation can be extended to model perimeter security at airports, which can encompass the security of terminals, lobbies, and entrances in airports. In this case, there is not a set of passengers or a set of flights. However, the number of people throughout the airport vary during the day, based on estimated arrival rates. Identifying performance measures for perimeter security problems can shed light on robust strategies for deploying surveillance equipment and screening devices at airports.

- In access control security systems, there are a number of checkpoints through which airport employees may pass. Designing systems that restrict unauthorized people from passing through access control checkpoints while allowing access to authorized airport employees can be used to provide an additional layer of security at the nation's airports.

- The Aviation and Transportation Security Act requires that cargo be screened for explosives. However, the Aviation and Transportation Security Act did not legislate specific screening guidelines or deadlines for cargo screening, and as a result, improvements in cargo screening have lagged well behind those for passenger and baggage screening. Extending the models for cargo screening can shed light on the optimal deployment and use of cargo screening devices.

- The models developed can be extended for screening passengers traveling on public transportation (e.g., buses, trains, and subways). In public transportation, there are a large number of stations where passengers can enter and exit the public transportation system, and the passengers who travel are unknown (i.e., passengers do not provide identification). Since many public transportation systems operate in cities with high population densities, developing models for public transportation security has the potential to protect city infrastructure in addition to passengers and transportation systems.

7.2 Conclusion

This dissertation introduces a systematic approach for designing and analyzing passenger screening systems using operations research methodologies. The issues discussed here represent but the tip of the iceberg. There are numerous problems in aviation security that can benefit from operations research methodologies, including improving perimeter access security, designing models for cargo screening, analyzing passenger throughput and space associated with security lines, and modeling secondary screening of passengers and their baggage when screening devices give an alarm response, to name just a few. By using operations research methodologies to gain insight into ways to improve aviation security system operations and performance, a lasting impression can be made on our nations security and well-being.

Bibliography

[1] V. L. L. Babu, R. Batta, and L. Lin. Passenger grouping under constant threat probability in an airport security system. *European Journal of Operational Research*, 168:633–644, 2006.

[2] A. Barnett. The worst day ever. *OR/MS Today*, 28(6):28–31, 2001.

[3] A. Barnett. CAPPS II: The foundation of aviation security? *Risk Analysis*, 24(4):909–916, 2004.

[4] A. Barnett, M. Abraham, and V. Schimmel. Airline safety: Some empirical findings. *Management Science*, 25(11):1045–1056, 1979.

[5] A. Barnett and M. K. Higgins. Airline safety: The last decade. *Management Science*, 35(1):1–21, 1989.

[6] A. Barnett, R. Shumsky, M. Hansen, A. Odoni, and G. Gosling. Safe at home? An experiment in domestic airline security. *Operations Research*, 49(2):181–195, 2001.

[7] J. F. Benders and J. A. E. E. van Nunen. A property of assignment type mixed integer linear programming problems. *Operations Research Letters*, 2(2):47–52, 1983.

[8] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, Cambridge, United Kingdom, 1998.

[9] E. A. Boyd. Polyhedral results for the precedence-constrained knapsack problem. *Discrete Applied Mathematics*, 41:185–201, 1993.

[10] R. A. Brualdi. *Introductory Combinatorics*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 1999.

[11] J. Bruno and P. Downey. Complexity of task scheduling with deadlines, set-up times and changeover costs. *SIAM Journal of Computing*, 7(4):393–404, 1978.

[12] V. Butler and R. W. Poole, Jr. Rethinking checked-baggage screening. Technical report, Reason Public Policy Institute, Public Policy No. 297, Los Angeles, CA, 2002.

[13] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123:333–345, 2000.

[14] P. Cartensen. The lamppost, the wizard and the law: Reflections on Professor Barnett's assessment of CAPPS II. *Risk Analysis*, 24(4):917–919, 2004.

[15] J. P. Caulkins. CAPPS II: A risky choice concerning an untested risk detection technology. *Risk Analysis*, 24(4):921–924, 2004.

[16] E. E. Chajakis and M. Guignard. Exact algorithms for the setup knapsack problem. *INFOR*, 32(3):124–142, 1994.

[17] S. Chakrabarti and A. Strauss. Carnival Booth: An algorithm for defeating the computer-assisted passenger screening system. *First Monday*, 7(10), 2002. Accessed at www.firstmonday.org on 23 October 2003.

[18] C. Derman, G. J. Lieberman, and S. M. Ross. A sequential stochastic assignment problem. *Management Science*, 18(7):349–355, 1972.

[19] L. Dugan, G. Lafree, and A. R. Piquero. Testing a rational choice model of airline hijackings. *Criminology*, 43(4):1031–1065, 2005.

[20] T. Frank. Checkpoint or choke point? *USA Today*, page 1A, 14 July 2005.

[21] G. H. Hardy, J. E. Littlewood, and G. Polya. *Inequalities*. Cambridge University Press, Cambridge, England, 1934.

[22] G. Heal and H. Kunreuther. IDS models of airline security. *Journal of Conflict Resolution*, 49(2):201–217, 2005.

[23] Dorit S. Hochbaum. A nonlinear knapsack problem. *Operation Research Letters*, 17:103–110, 1995.

[24] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.

[25] O. H. Ibarra and C. E. Kim. Approximation algorithms for certain scheduling problems. *Mathematics of Operations Research*, 3(3):197–204, 1978.

[26] G. P. Ingargiola and J. F. Korsh. A general algorithm for the one-dimensional knapsack problem. *Operations Research*, 25:752–759, 1977.

[27] S. H. Jacobson, J. M. Bowman, and J. E. Kobza. Modeling and analyzing the performance of aviation security systems using baggage value performance measures. *IMA Journal of Management Mathematics*, 12(1):3–22, 2001.

[28] S. H. Jacobson, T. Karnani, and J. E. Kobza. Assessing the impact of deterrence on aviation checked baggage screening strategies. *International Journal of Risk Assessment & Management*, 5(1):1–15, 2005.

[29] S. H. Jacobson, J. E. Kobza, and A. S. Easterling. A detection theoretic approach to modeling aviation security problems using the knapsack problem. *IIE Transactions*, 33:747–759, 2001.

[30] S. H. Jacobson, L. A. McLay, J. E. Kobza, and J. M. Bowman. Modeling and analyzing multiple station baggage screening security system performance. *Naval Research Logistics*, 52(1):30–45, 2005.

[31] S. H. Jacobson, L. A. McLay, J. L. Virta, and J. E. Kobza. Integer program models for the deployment of airport baggage screening security devices. *Optimization and Engineering*, 6(3):339–359, 2005.

[32] S. H. Jacobson, J. E. Virta, J. M. Bowman, J. E. Kobza, and J. J. Nestor. Modeling aviation baggage screening security systems: A case study. *IIE Transactions*, 35(3):259–269, 2003.

[33] D. S. Johnson and K. A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.

[34] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 83–103. Plenum Press, New York, 1972.

[35] H. Kellerer and U. Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *Journal of Combinatorial Optimization*, 3:59–71, 1999.

[36] H. Kellerer and U. Pferschy. Improved dynamic programming in connection with an FPTAS for the knapsack problem. *Journal of Combinatorial Optimization*, 8:5–11, 2004.

[37] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, Berlin, 2004.

[38] J. E. Kobza and S. H. Jacobson. Addressing the dependency problem in access security system architecture design. *Risk Analysis*, 16(6):801–812, 1996.

[39] J. E. Kobza and S. H. Jacobson. Probability models for access security system architectures. *Journal of the Operational Research Society*, 48(3):255–263, 1997.

[40] H. Kunreuther and G. Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26(2):231–249, 2003.

[41] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.

[42] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report No. 178, Computer Science Laboratory, Princeton University, Princeton, NJ, 1975.

[43] M. J. Magazine and O. Oguz. A fully polynomial approximation algorithm for the 0-1 knapsack problem. *European Journal of Operational Research*, 8:270–273, 1981.

[44] S. Martello and P. Toth. Branch and bound algorithms for the solution of general unidimensional knapsack problem. In *Advances in Operations Research*, pages 295–301. Plenum Press, North Holland, Amsterdam, 1977.

[45] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley & Sons, New York, NY, 1990.

[46] S. Martonosi and A. Barnett. Security profiling of airline passengers: How effective would it be? INFORMS National Meeting, October 19-22, 2003, Atlanta, GA, 2003.

[47] S. E. Martonosi. *An Operations Research Approach to Aviation Security*. PhD thesis, Massachusetts Institite of Technology, Cambridge, MA, 2005.

[48] L. A. McLay and S. H. Jacobson. Integer knapsack problems with set-up weights. *Computational Optimization and Applications*, to appear, 2007.

[49] L. A. McLay, S. H. Jacobson, and John E. Kobza. Making skies safer: Applying operations research to aviation passenger prescreening systems. *OR/MS Today*, 32(5):24–31, 2005.

[50] L. A. McLay, S. H. Jacobson, and John E. Kobza. When is selective screening effective for aviation security? Technical report, University of Illinois, Urbana, IL, 2005.

[51] L. A. McLay, S. H. Jacobson, and John E. Kobza. Integer programming models and analysis for a multilevel passenger screening problem. *IIE Transactions*, to appear, 2006.

[52] L. A. McLay, S. H. Jacobson, and John E. Kobza. A multilevel passenger prescreening problem for aviation security. *Naval Research Logistics*, 53(3):183–197, 2006.

[53] K. M. Mead. Deployment and use of security technologies. *Office of Inspector General, Department of Transportation, Washington, D.C.,*, Report Number CC-2002-007, 2002.

[54] K. M. Mead. Key issues concerning implementation of the Aviation and Transportation Security Act. *Office of Inspector General, Department of Transportation, Washington, D.C.*, Report Number CC-2002-098, 2002.

[55] K. M. Mead. Aviation security costs, Transportation Security Administration. *Office of Inspector General, Department of Transportation, Washington, D.C.*, Report Number CC-2003-066, 2003.

[56] L. Miller. Delta to test new airport security plan. *Associated Press*, 28 February 2003. Accessed at news.yahoo.com on 28 February 2003.

[57] National Research Council. *Airline Passenger Security Screening.* National Academy Press, Publication NMAB-482-1, Washington, D.C., 1996.

[58] National Research Council. *Assessment of Technologies Deployed to Improve Aviation Security.* National Academy Press, Publication NMAB-503, Washington, D.C., 2002.

[59] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, 1988.

[60] Official Airline Guide. *OAG Business Travel Planner: North American Edition.* Official Airline Guides, Bedfordshire, UK, 1998.

[61] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., New Jersey, 1982.

[62] K. Park and S. Park. Lifting cover inequalities for the prcedence-constrained knapsack problem. *Discrete Applied Mathematics*, 72:219–241, 1997.

[63] U. Pferschy. Dynamic programming revisited: Improving knapsack algorithms. *Computing*, 63(4):419–430, 1999.

[64] U. Pferschy, D. Pisinger, and G. J. Woeginger. Simple but efficient approaches for the collapsing knapsack problem. *Discrete Applied Mathematics*, 77:271–280, 1997.

[65] D. Pisinger. A fast algorithm for strongly correlated knapsack problems. *Discrete Applied Mathematics*, 89:197–212, 1998.

[66] D. Pisinger. A minimal algorithm for the bounded knapsack problem. *INFORMS Journal on Computing*, 12(1):75–82, 2000.

[67] R. W. Poole, Jr. and George Passantino. A risk-based airport security policy. Technical report, Reason Public Policy Institute, Public Policy No. 308, Los Angeles, CA, 2003.

[68] W. B. Powell. Approximate dynamic programming for asset management: Solving the curses of dimensionality. Technical report, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, 2004.

[69] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

[70] I. Ravid. Safety versus defense: Comments on "CAPPS II: The foundation of aviation security?". *Risk Analysis*, 24(4):929–931, 2004.

[71] J. D. Rhodes. CAPPS II: Red light, green light, or mother, may I?. *Journal of Homeland Security*, Accessed at www.homelandsecurity.org on 7 September 2005, 9 March 2004.

[72] S. M. Ross. *Introduction to Probability Models*. Harcourt Academic Press, San Diego, 7th edition, 2000.

[73] H. Sural, L. N. van Wassenhove, and C. N. Potts. The bounded knapsack problem with setups. Report 97/71/TM, INSEAD, Centre for the Management of Environmental Resources, Cedex, France, 1997.

[74] B. Toktas, J. W. Yen, and Z. B. Zabinsky. Addressing capacity uncertainty in resource-constrained assignment problems. Technical report, University of Washington, Seattle, WA, 2003.

[75] B. Toktas, J. W. Yen, and Z. B. Zabinsky. A stochastic programming approach to resource-constrained assignment problems. Technical report, University of Washington, Seattle, WA, 2004.

[76] United States Department of Homeland Security, Office of the Press Secretary. Homeland Security Presidential Directive-3. Press Release, March 12 2002.

[77] United States Federal Aviation Administration. FAA aerospace forecasts fiscal years 2006–2017. Report, Office of Policy and Plans, Washington, D.C., 2002.

[78] United States Government Accountability Office. Secure Flight development and testing under way, but risks should be managed as system is further developed. Technical Report GAO-05-356, Washington, D.C., March 2005.

[79] United States House of Representatives. Aviation security with a focus on passenger profiling. Hearing memo, 107th congress, Committee on Transportation & Infrastructure, Subcommittee on Aviation, Washington, D.C., 27 February 2002.

[80] United States Transportation Security Administration. TSA's CAPPS II gives equal weight to privacy, security. Press release, Department of Homeland Security, Washington, D.C., 11 March 2003.

[81] United States Transportation Security Administration. Twelve explosives detection canine teams join TSA/Homeland Security. Press release, Department of Homeland Security, Washington, D.C., 14 November 2005.

[82] United States Transportation Security Administration. TSA unveils enhanced security screening procedures and changes to the prohibited items list. Press release, Department of Homeland Security, Washington, D.C., 2 December 2005.

[83] United States Transportation Security Administration. TSA announces key elements of Registered Traveler Program. Press release, Department of Homeland Security, Washington, D.C., 20 January 2006.

[84] United States Transportation Security Administration. FAMS mission and history. Security & law enforcement, Department of Homeland Security, Washington, D.C., 2006.

[85] United States Transportation Security Administration. "Secure Flight" to be tested before year's end. Press release, Department of Homeland Security, Washington, D.C., 26 August 2004.

[86] United States Transportation Security Administration. TSA continues to roll-out explosives detection trace portals. Press release, Department of Homeland Security, Washington, D.C., 27 July 2005.

[87] United States Transportation Security Administration. Personal communications, TSA briefing, Arlington, VA, 3 October 2003.

[88] J. E. Virta, S. H. Jacobson, and J. E. Kobza. Outgoing selectee rates at hub airports. *Reliability Engineering and System Safety*, 76(2):155–165, 2002.

[89] J. E. Virta, S. H. Jacobson, and J. E. Kobza. Analyzing the cost of screening selectee and non-selectee baggage. *Risk Analysis*, 23(5):897–908, 2003.

[90] W. H. Yang and C. J. Liao. Survey of scheduling research involving setup times. *International Journal of Systems Science*, 30(2):143–155, 1999.

Author's Bibliography

Laura was born on July 13, 1978 to James and Julia Albert in Schaumburg, Illinois. After vowing never to pursue a career in the sciences, Laura was admitted to the University of Illinois at Urbana-Champaign as an undergraduate history major. After her Freshman year, Laura saw the error of her ways and transferred to the College of Engineering, where she received her B.S. and M.S. in General Engineering in 2000 and 2001. As an undergraduate, Laura was a Chancellor's scholar, graduated with honors, and received an International minor in German.

While studying the design of aviation security systems under the guidance of Professor Sheldon H. Jacobson, Laura has had nine journal publications and has presented her research at several conferences. Her dissertation has been supported in part by the Department of Mechanical & Industrial Engineering Alumni Board Teaching Fellowship and the Arms, Control, Disarmament, and International Security Thesis Initiation Fellowship. She also received the Department of Mechanical & Industrial Engineering George W. Harper award for excellence in safety engineering.

Laura has had a number of non-academic adventures while in graduate school. She married the love of her life, Courtlan McLay, in December 2002, and their daughter, Eleanor, was born in October 2004. When she's not chasing Eleanor, Laura runs marathons, plays volleyball, sews, and bakes. Laura will join the Department of Statistical Sciences & Operations Research at Virginia Commonwealth University in Richmond, Virginia as an Assistant Professor.