THE WARTIME PORTFOLIO SELECTION PROBLEM

by

Ronald F. A. Woodaman
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Systems Engineering and Operations Research

Committee:

_____	Dr. Karla L. Hoffman, Dissertation Director
_____	Dr. Andrew G. Loerch Committee Member
_____	Dr. Ariela Sofer Committee Member
_____	Dr. Kenneth Hintz Committee Member
_____	Dr. Ariela, Sofer, Department Chair
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date:_____	Spring Semester 2015 George Mason University Fairfax, VA

The Wartime Portfolio Selection Problem

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Ronald F. A. Woodaman
Master of Science
Naval Postgraduate School, 2000
Bachelor of Science
United States Naval Academy, 1987

Director: Karla Hoffman, Professor
Department of Systems Engineering and Operations Research

Spring Semester 2015
George Mason University
Fairfax, VA

DEDICATION

To the Creator who made and blessed me.

To my parents who loved me and raised me.

To my wife who shares her life with me.

To my children who fulfill me.

To the men and women who served our Nation with me.

ACKNOWLEDGEMENTS

I wish to acknowledge the contributions of my dissertation director Dr. Karla Hoffman. I think I echo her many students in saying that she goes well beyond the norm in maximizing the value of every dissertation under her supervision.

I want to thank Dr. Andrew Loerch, who has been a mentor and guide to me along this way. So much of what I have learned in the practice of military operations research came via his instruction. Many of his practical and insightful ideas found this way into my dissertation, particularly the DENIV approach.

I want to thank Dr. Ariela Sofer, my department chair, both for serving on my committee and the outstanding instruction I got from her in the classroom. She developed my appreciation for the fundamentals of optimization: an improving direction and a step length.

I wish to also recognize Dr. Kenneth Hintz for serving on my committee. While working with him on the JIEDDO project he inculcated in me a profound respect for the value of intellectual property.

In mentioning the JIEDDO project, I must also acknowledge Dr. Kathryn Laskey both for her encouragement of my research, and for her outstanding leadership of the team.

Two of the key technologies in this dissertation were taught to me by two outstanding professors. I learned stochastic programming from Dr. Miguel Lejeune of George Washington University, and dynamic programming from Dr. Rajesh Ganesan of our SEOR Department. My thanks to both of these passionate teachers.

Thanks to my father, R. Elliott H. Woodaman, who earned his PhD from Georgetown University, who from an early age taught me never to quit on your dreams, and to my mother, Nivia Luz Woodaman, who taught me that is everything is possible with Love.

Thanks to my darling children Zoe, Iain, and Isabella, for whom writing this dissertation just seemed to be an excuse for their father to miss one event or another.

Lastly, thanks to my lovely wife Michelle, without whom this would never have been possible, for her enduring love, patience, and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

ABSTRACT

THE WARTIME PORTFOLIO SELECTION PROBLEM

Ronald F. A. Woodaman, PhD

George Mason University, 2015

Dissertation Director: Dr. Karla Hoffman

This thesis describes the research conducted to support the optimal selection of a portfolio of military solutions during wartime. During peacetime, the United States military selects a portfolio of military solutions holistically as part of an annual budgetary planning cycle supported by long-term planning. During wartime, this annual review and decision process is not responsive to the rapid cycles of battlefield adaptation and the resulting exploitation of opposing capability gaps by adversaries. The long-running vulnerability of U.S. forces in Iraq and Afghanistan to Improvised Explosive Device (IED) attacks, as the threat's tactics and techniques evolved during these conflicts, provides a poignant example. During conflict, opportunities to improve the force arrive irregularly over time and are difficult to anticipate. When potential solutions are identified, these must be rapidly pursued, subject to resource constraints. Bad decisions

early rob resources from better opportunities that arrive later, while good opportunities unduly delayed may lead to lost opportunities on the battlefield.

We develop quantitative methods to support decision makers in the optimal selection of solutions in this context, employing as our motivating case study the challenges faced by the United States Department of Defense's Joint Improvised Explosive Device Defeat Organization (JIEDDO). For example, in fiscal year 2013, the JIEDDO budget was $1.6 billion. This organization had to make counter-IED solution selection decisions continuously as these arrived, without knowing precisely what other opportunities might occur in the future months. Two key aspects of this problem are how to measure the military value of potential solutions, and how to make the best choices as opportunities arrive. Correspondingly, this dissertation examines these two sides: valuation of war-fighting solutions in an uncertain and time-sensitive context, and, given a valuation method, methods for optimal sequential selection.

CHAPTER 1 – INTRODUCING THE PROBLEM

1.1 Introduction

During peacetime, the United States military seeks to optimize its portfolio of capabilities via a multi-year process of planning, budgeting, and review during which varying requirements compete for limited funds. These requirements include manpower, equipment, supplies, and the facilities to house them. The planning must address that for much of the equipment, the lifetime between inception and retirement can exceed a half century, a span during which the government funds the research and development, acquisition, and disposal. This broadly describes the standard Military Capital Planning Problem (MCPP) (see Brown et al, 2007).

During short conflicts, such the Battle of the Falklands in 1983, the invasion of Panama in 1989, and the Russian-Georgian conflict 2008, a military must fight the entire conflict with its starting inventory of war-fighting equipment, adequate or not. If the conflict lasts longer, a country's military may have the opportunity to address inventory inadequacies of two types: new approaches to exploit an adversary's weaknesses and new ways to shore up its own vulnerabilities. In particular, these inadequacies may exist in a particular asymmetric domain of the conflict.

During World War II, the Allies' dominance of the Atlantic via their superior naval forces was severely contested by the Nazi U-boat fleet. This led to the accelerated development of anti-submarine technologies and tactics, and to the Allies' eventual success in the Battle of the Atlantic (Morison, 1963). More recently this asymmetry existed between Allied forces and their enemies in Iraq and in Afghanistan in the form of the Improvised Explosive Device (IED).

The same long-term, measured acquisition processes that support an inventory optimization approach during peacetime may be inadequate to keep pace with the cycle of adaptation on the battlefield. Defense organizations need an alternative approach that is more responsive to the demands of war. How has the U.S. adapted to this challenge of war-time acquisition?

1.2 The Joint Improvised Explosive Device Defeat Organization

A U.S. Defense organization specifically created to address this challenge is the Joint Improvised Explosive Defeat Organization (JIEDDO). JIEDDO coordinates and supports activities across all of the Military Services aimed at defeating the Improvised Explosive Device (IED) as a weapon of strategic influence. The conflicts in Iraq and Afghanistan exposed the inadequacy of the U.S.'s inventory in the realm of irregular and asymmetric warfare, in particular the vulnerability of its tactical vehicles to IEDs. JIEDDO provides the regional Combatant Commanders – the leaders in charge of the theaters of war - with a centralized clearing-house for requesting and acquiring counter-IED capabilities.

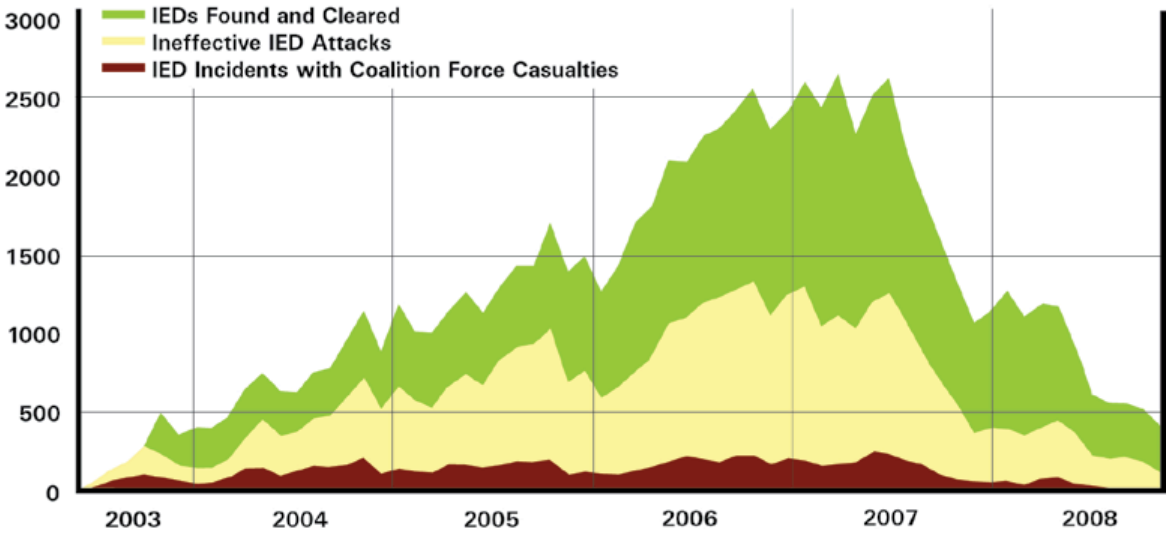

Figure 1-1 IED Incidents against Coalition Forces in Iraq (JIEDDO Annual Report FY08)

Figure 1 shows the historical level of IED activity in Iraq from 2003 through 2008. Between Fiscal Years (FY) 2004 and 2008, JIEDDO received $13.79B in appropriations (2008 HASC Report), the bulk of which was allocated to procuring counter-IED initiatives. These initiatives cover the gamut of military operations and functions: software, unmanned aircraft systems, an overwhelming variety of sensing technologies, better armored vehicles, training programs, and many kinds of specialized support programs. The initiatives funded by JIEDDO may have played a part in keeping the number of monthly casualty-causing IED attacks (the red area on the graph) somewhat flat relative to the great spikes in the total number of IEDs incidents until the steep drop in overall activity during 2007.

The next figure shows how IED incidents in Afghanistan eventually surpassed those in Iraq and contrasts these trends with JIEDDO's annual budgets during this time.

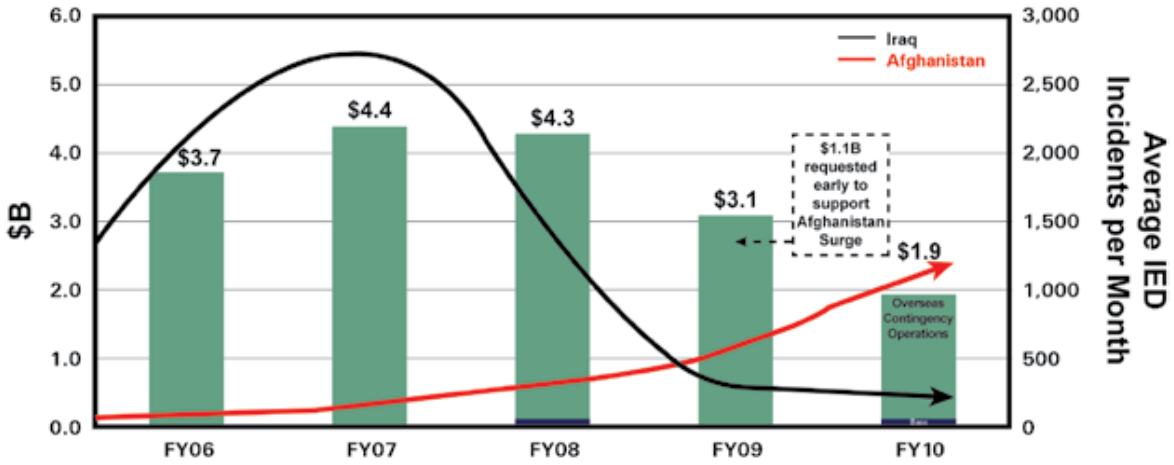

Figure 1-2 IED incidents per month in Iraq and Afghanistan contrasted with JIEDDO's annual budgets (Figure from the JIEDDO Annual Report FY2010)

Because JIEDDO presents a very specialized – and large – example of the wartime problem of dynamically selecting the best mix of solutions, we will use JIEDDO throughout this dissertation as a motivational case study.

1.3 The Challenges of Accelerated Acquisition

To rapidly meet emerging needs, JIEDDO inverts the standard MCPP. Instead of a drawn-out process of winnowing down an "optimal" subset from a list of military investment choices, JIEDDO receives its budget up front from Congress without certainty as to what will be procured and what will be deployed eventually. Then, as possible counter-IED solutions – "initiatives" - present themselves or are discovered over time, JIEDDO is mandated to rapidly develop, test, and acquire the best available initiatives in order to expedite useful solutions to the battlefield.

Unnecessarily delayed acquisition of effective counter-IED solutions can lead to unnecessary loss of life and failed military objectives. But funding ineffective solutions – whether from poorly understood technologies, engineering management, random outcomes, or changes in operational requirements – may deny funds to effective solutions arriving later as well as consuming limited non-fungible resources such as weapons test facilities, developmental laboratories, and management time. To manage the process of rapidly identifying, investing, developing, testing, and procuring promising technologies, JIEDDO employs the JIEDD Capability Acquisition and Approval Process (JCAAMP).

JIEDDO has faced criticism from the Government Accountability Office (GAO Report 2007), from Congressional inquiries (House Armed Services Committee report 2008), and from military personnel themselves (Ellis et al, 2007). One recurring theme is JIEDDO's inability to clearly specify measures that relate to goal attainment, how these map to the initiatives it funds, and the rationale for its funding choices.

From studying JIEDDO's particular investment selection problem, we hope to develop some principles that others may apply to the more general problem of wartime defense portfolio selection.

1.4 Wartime Portfolio Selection Problem Description

In this section we will present a general problem description for the War Time Portfolio Selection. To aid in developing this problem statement, we consider the more specific case of JIEDDO's particular challenge.

1.4.1 JIEDDO Decisions When Selecting Initiatives

We summarize the basic decisions involved in JIEDDO search for the best portfolio of counter-IED initiatives:

- JIEDDO searches for promising counter-IED initiatives to rapidly acquire and field in order to save lives and defeat the enemy.
- JIEDDO receives annual funding increments in order to select the most promising amongst potential solutions that arrive at random intervals throughout the year.
- After arriving, initiatives are rapidly evaluated for suitability against a combination of stated needs from theater prior to an initial funding decision. No single measure of value exists to support return on investment (ROI) or cost-benefit analysis.
- JIEDDO chooses whether to fund each initiative based a variety of factors, to include the priority of the need as described by commanders in the theater of war[1], its own technology assessments, input from the military services, and the funds available.
- If an initiative is selected, JIEDDO funds it in successive stages: initial development of initiatives; various demonstrations phases, culminating in the theater of conflict; and the sustainment of initiatives.

[1] The standard communication of need by the Combatant Commander is the Joint Universal Operational Needs (JUONS) as described in the Chairman of the Joint Chiefs of Staff Instruction 3740.01 *Rapid Validation and Resourcing of Joint Urgent Needs.*

- JIEDDO funds the initiatives for no more than a total of 2 years, after which the items is either turned over to one of the Military Services or it is terminated and disposed.

Given the urgency of the counter-IED (C-IED) fight, JIEDDO needs to identify, develop, demonstrate and sustain the most effective portfolio of C-IED initiatives subject to budget constraints. Choices are difficult because there is no certainty that an initiative will function as specified, that it can deployed in a timely fashion, or that a better choice might not arrive shortly.

1.4.2 Random Initiative Arrivals and their Costs

JIEDDO, via various methods, conducts an active search for solutions. From an enterprise perspective, these solutions "arrive" over time in an unpredictable fashion. A preliminary data analysis of approximately 18 months of funded initiative data provided some of the following insights on the rate at which initiatives arrive and a description of the costs of some of these.

Figure 1-3 shows the number of new initiatives funded each week between 3/21/07 and 9/21/08, the period of time our research team had access to these data[2]. (Note: The poor quality of the image is the result of transferring this unclassified image from the classified network domain in which it was developed. This applies to the figures on the following pages as well.)

[2] For obvious reasons, we would prefer to have the entire set of arriving initiatives – not just the funded ones - and be able to describe the distribution of costs for all arriving initiatives. When these data were collected, JIEDDO had not been keeping this information on the non-selected initiatives.

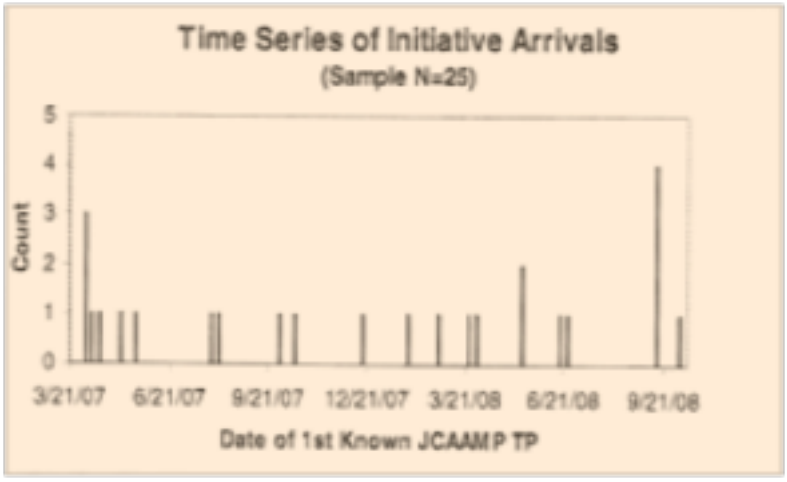

Figure 1-3 Number of initiatives funded weekly between 03/21/07 and 9/21/08

Visually, these data appear random. Were we to model a decision process based on this particular observed behavior, an approach might be to model this as a random arrival process. A simple random arrival process, found throughout operations research literature, is the Poisson random arrival process.

A Poisson process $\{t, N(t)\}$ counts the number of events from the time 0 to time t. Ross (1997) provides the following definition:

- $N(0) = 0$;
- independent increments – the number of arrivals in one time increment do not depend upon the arrivals in a different time increment;
- stationary increments – the distribution of the number of arrivals in a time period depend only on the length of the time period;

- the number of arrivals in a time interval follow a Poisson distribution;
- the time interval between discrete arrivals follows an Exponential distribution;

The Poisson distribution has the feature that the variance equals the mean. Therefore, a gross check of the adequacy of the Poisson distribution as a model of a data set is to compare the sample mean with the sample variance. For our sample of funded initiatives, the average number of weekly arrivals was 0.3205 and the sample variance was 0.4804. Therefore, the Variance-to-Mean Ratio (VMR) for this data set is 0.4804/0.3205 = 1.499. A VMR less than 1 indicates under dispersed data (more clustering around the mean than expected for Poisson data), and a number greater than 1 indicates overly dispersed data. Thus, the sample is overly dispersed with more observations in the lower and the higher cells and fewer in the central cells than expected for Poisson data.

Statistical hypothesis tests can be applied to evaluate whether deviations from a theoretical distribution indicate inadequacy of the hypothesized distribution, or whether they can be explained as chance fluctuations. The chi-squared test is one of the most commonly applied tests. For small samples, a common adjustment made to the chi-squared test is the Yates correction. For this data set, we calculated the chi-squared statistic with Yates correction, comparing against the Poisson distribution, and obtained a p-value of 0.0364. Without the Yates correction, the standard chi square goodness of fit test provides a p-value of 2.3e-8, a very small number.

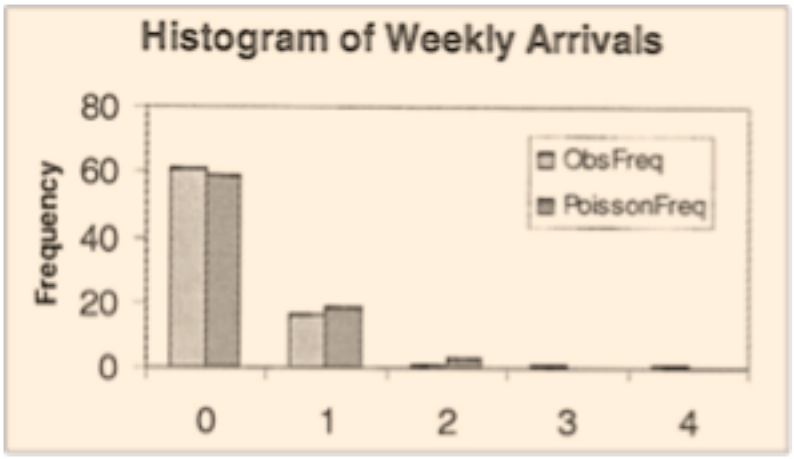

Figure 1-4 Comparing weekly arrivals to a Poisson distribution with the same mean

Figure 1-4 provides a histogram comparing the distribution of arrivals to the distribution of Poisson arrivals with the same mean. The main issue lies in a larger right tail in the observed sample than one would expect in the Poisson distribution of same mean, specifically the two weeks, one with three arrivals and the other with four.

We frequently observed that scheduling challenges resulted in the weekly decision meeting being rescheduled to the following week. This occurred around 20% of the time. Consequently, any arrivals from that week were viewed the following week. Thus, while the arrival process might be behaving closer to the Poisson ideal, the decision process itself did not exactly follow the ideal. We did not witness every such meeting and so cannot attest to whether the case of the four arrivals in one week was the result of such a rescheduling. Removing the four-arrival week without substitution makes the goodness-of-fit for the unadjusted chi-square p-value jumps from 2.3e-8 to 0.44. It may

well be that the inconsistency in the weekly decisions led to this skew in the observed weekly arrivals.

Save for this one, the Poisson arrival process closely mimics the quality of the observed behavior. Thus, for modeling purposes, we will assume that arrivals follow this Poisson distribution.

The next question we can ask of the data concerns the distribution of funded initiative costs. The costs we observed ranged generally from 100 thousand to one billion. Using the same time period as Figure 2, Figure 4 shows the same time series of funded initiatives in a different way. Instead of the counts of funded initiatives, this chart shows the aggregate costs of funded initiatives in each week.

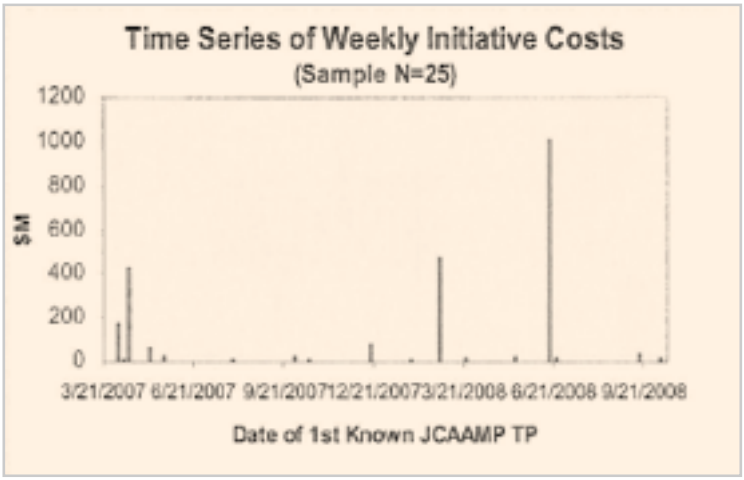

Figure 1-5 Total aggregate weekly cost of funded initiatives between 3/21/07 and 9/21/08

It has been long observed that U.S. government agencies' spending patterns show a surge at the end of the fiscal year when the spending authority is about to expire (GAO,

1980). To combat this tendency, Congress funds JIEDDO with money that does not expire at the fiscal year. In this time series there does not appear to be a pattern of end of year spending.

To facilitate analysis, we are interested in a distribution that may fit these data, in particular given the many orders of magnitude present in the data. Figure 5 shows a histogram of the $\log_{10}$-transformed costs.

Figure 1-6 Comparing the distribution of $\log_{10}$ transformed costs to the Normal distribution with same mean and standard deviation

Based on this second histogram, the costs of past funded initiatives appear to be log-normally distributed, with the mean in $\log_{10}$-space being 7.20 and standard deviation of 0.82. A chi-squared test comparing the transformed data with a normal distribution of the same parameters defends this hypothesis, with a p-value of 0.4616. The $\log_{10}$-mean and ±1 standard deviation range in log-space equate to \$15.8M and [2.4M, 104.7M]

respectively. We should expect that about 68% of funded initiative costs (rounding to nearest million) to fall between $2M and $105M.

While $\log_{10}$ provided an intuitive basis for understanding the cost behavior, further research into the log-normal showed that most formulae employ the natural logarithm. For these data the log-mean μ comes to 16.58 and the variance σ^2 to 3.57. For the normal distribution, the mean μ equals the median. Transforming from log-space back to linear space log-median translates to e^{μ}; but this is not true for the mean. The log-normal distribution's pronounced right tail pulls the mean to the right, so the mean of the log-normal distribution has the following form: $e^{\mu+\sigma^2/2}$. This equates to a mean of $93.4M for the distribution, which is very close to the sample mean of $93.6M and far higher than the median of $16M.

Combining these observations provides us with a basic model for the funding decisions over time: *funded counter-IED initiatives behave as a compound arrival process, where the number of arrivals in a given week is Poisson distributed, and the costs of these initiatives are log-normally distributed.* A missing part of this model is how to measure the value, or benefit, of an initiative.

Assuming that these observations can be generalized and extended to other like organizations focused on the war time acquisition of critical capabilities, we use the information gleaned to make a general problem statement.

1.5 Wartime Portfolio Selection Problem (WPSP) Statement

A defense acquisition agency has the mission to rapidly acquire capabilities within a critical domain to support an active and enduring war. The agency starts the funding period with a budget of discretionary funds and a portfolio of previously funded initiatives. Initiatives are funded efforts that, with development and testing, may yield war-fighting capabilities. Throughout the year, the agency seeks to add initiatives that will improve its portfolio. Initiatives arrive via a random process. When an initiative is added to the portfolio the budget is decremented by the first year cost of the initiative, which is not known until the initiative arrives.

Initiatives have a random benefit that cannot be ascertained until they arrive, is measured in terms of their potential contribution to the current domain portfolio, and is non-fungible. Initiatives must be accepted or rejected in short order, as not adding a "good" initiative when it presents itself could result in the loss of a potential war-winning opportunity. Conversely, spending a lot of money on a "bad" initiative could result in lost purchasing power and developmental resources. The opportunity to assess newly arrived initiatives, and the decision whether to fund or reject, happens at fixed time intervals (e.g., weekly, monthly).

Funded initiatives receive a year of funding, which may be used to complete development and testing, acquire the requisite amount of capability, deploy the capability, and provide life-cycle support. Initiatives that prove themselves useful

capabilities may receive funding – but this decision is outside of the scope of this problem statement.

The agency has imperfect knowledge of what initiatives may arise during the course of the year. It must make its decisions sequentially.

1.6 Research Questions

The classic operations research resource allocation problem is the knapsack problem. Given a knapsack of limited capacity and set of items with different benefits and size, one must choose the subset with the most benefit that fits in the knapsack.

The wartime portfolio problem differs in key respects: the items are random in quantity, present themselves sequentially, their benefits and size (cost) are not known until they arrive, and given the urgency of the situation, the decisions must be made sequentially as soon as initiatives arrive. At a deeper level, we have to consider that initiatives are funded not via a single decision but rather are funding in stages with their own funding increments. Promising initiatives may prove unattractive after further testing, due to failures in performance or unacceptable cost growth. We may also want to consider "side" constraints – constraints that might either require investment in some capability domains or limit investment in other domains regardless of value.

We will focus on two research questions:

RQ1. How might we measure the benefit of the initiatives, particularly since we wish to use this information to help us to decide which initiatives to fund?

RQ2. If we had the means to measure their benefit and cost, how might we optimally choose among these randomly arriving initiatives?

Chapter 2 provides an overview of the applicable literature we have identified. Chapter 3 describes how we model the benefit of initiatives in the case of a counter-IED portfolio and draw some extensions to the larger wartime portfolio problem. Chapter 4 examines an approach based on Dynamic Programming. Chapter 5 describes a different approach to the problem employing Stochastic Integer Programming. Chapter 6 summarizes the work and discusses further research.

CHAPTER 2 – LITERATURE REVIEW

2.1 Introduction

This literature review examines two aspects. We first review literature on related optimization and mathematical programming approaches that may be of utility in our research into the WPSP. The second part of the reviews deals with research concerning means of measuring the value of a military portfolio.

2.2 Applicable Mathematical Programming Literature

The knapsack problem (KP), while simple in structure, has been studied extensively throughout operations research and related fields. It appears in transportation, capital planning, communications, military applications, agriculture, etc.

The KP concerns a situation in which one must choose items from a set of items, where each item has a benefit, value, or utility, and a weight, cost, or capacity. The knapsack has finite capacity, assumed to be less than the aggregate capacity of the set of items. The question is which combination of items to choose to maximize to total benefit. The linear knapsack problem assumes additivity of benefits and weights.

Variations on the deterministic version of the KP – where all data are known at the time a decision must be made – include: the 0-1 knapsacks, where the choices are to either include an item or not; the bounded knapsack, where one can include multiple copies of an item up to some bound; and the unbounded knapsack. Early exact solution

approaches include *dynamic programming* (Bellman, 1957) and *branch-and-bound* (Kolesar, 1967). An early but still useful heuristic approach is the *greedy approximation algorithm* (Dantzig, 1957). Salkin and de Kluyver (1975) provide a survey of dynamic programming, integer programming, lagrangian-based heuristics, and network based approaches. Wilbault, Hanafi, and Salhi survey a variety of innovations in heuristic approaches largely focused on the deterministic variants of the (2008).

We are concerned with the stochastic version the problem – where some of the data are uncertain. Now instead of a known benefit or weight, we have random variables describing benefits, weights or both. We will examine various approaches and see how these may relate to the JPSP.

We start with the concept of portfolio optimization. Markowitz (1952) models the optimization of portfolios of financial assets as the quadratic minimization of covariance (risk) subject to some level of expected return. The repeated solution of the model for different levels of return creates what Markowitz (1959) called the critical line. Portfolios below this line are inefficient and those above cannot exist. Along this frontier, improving the expected return cannot happen without an increase in risk nor can reducing risk be accomplished without reducing the level of return. The knapsack constraint is of the unbounded variety since, subject to the total amount of funds, it assumes that one can buy any amount of stock. The distribution of stock returns is assumed Gaussian.

During this similar time-frame, Bellman examined sequential decision problems. He noted that a broad class of these displayed a similar decomposable structure, where if one could solve the sub-problem, then via recursion, the entire problem could be solved. Bellman called this concept of solving the sub-problem the Principle of Optimality. Thus, was born Bellman's *dynamic programming* (1957). We provide a more thorough review of dynamic programming in Chapter 4.

Markov Decision Processes (MDP) describes a group of sequential decision problems where the outcome of the decision is random (Denardo, 2003). The process is memory-less in that the system's current state provides all the necessary information about the system. At each time step a decision must be made and the system's transition probabilities depend upon the decision. Transitions from one state to another provide a certain reward. The goal is to find the policy that maximize the accumulation of discounted reward. Bellman (1957) described stochastic decision problems that were essentially MDPs. Howard (1960) brought the term into wide-spread use and devised *policy iteration* as a solutions approach.

Dantzig (1955), who had already engendered the field of linear programming, examined what he called two-stage linear programs with uncertainty. He describes these as problems where there were certain data and decisions to be made in the present, random data to be revealed at a future juncture, recourse decisions that could made subsequent to this revelation, and where the decision maker seeks to optimize an objective function that incorporates both deterministic and random elements.

Independently, Beale (1955), examined linear programs with random coefficients and proved that these were convex. Together these two papers formed the start of stochastic programming.

Cord (1963) looks at choosing amongst capital investment projects internal to a firm. Now the decision variable is binary instead of continuous non-negative; a project will either be funded or not. In contrast to stock portfolios, he assumes independence of returns, arguing that projects with interdependencies will be mutually exclusive, and that the choice between these will have already been made. He describes a decision rule where projects are chosen in order of interest rate subject to funds available and the weighted average of the individual variances of the candidates for the portfolio. This creates two knapsack constraints. He solves the problem as a dynamic program, where each stage is the choice of a project, and within each stage the budget is varied incrementally from 0 to the total amount available. He addresses the variance constraint via a Lagrange multiplier. His approach only works for budget problems with a single budgetary period.

Greenberg (1968) broaches the subject of the stochastic knapsack (which he calls a dynamic program with linear uncertainties) where either the utilities or weights of the knapsack are random variables. For the case of stochastic weights, he converts this to a *chance constraint* (Charnes and Cooper, 1963): where the linear combination of the weights must satisfy the total capacity with probability p. Assuming Gaussian weights, this chance constraint can be restated as a deterministic constraint. In the case of

stochastic utilities, he employs a chance objective function, maximizing total utility for given p. He solves both approaches via dynamic programs with two-state variables. Greenberg's problem is what Birge and Louveaux (1997) describe as a *static stochastic program*; there is no recourse after the decision is made and uncertainties revealed.

Henig (1989) examines a similar problem. He argues against considering the problem of chance constraints, since he states that in most applications, the weights are known at the time of the decision. The uncertainty – again Gaussian - surrounds the returns. As his objective function he uses a convex combination of mean and variance. He points out that Greenberg's method is not practical for realistic problems. He uses dynamic programming and experiments with nested search procedures to keep his state space manageable.

Morita et al (1989) treat a similar problem starting with an objective function that maximizes the probability of exceeding a threshold. The resulting deterministic objective function leads to a non-linear fractional program formulation and uses methods outlined by Dinkelbach (1967).

Morton and Wood (1998) examine a similar problem of fixed weights and random rewards, where the objective is to maximize the probability that a given linear combination of random rewards exceeds a threshold. They compare dynamic programming and integer programming approaches and show that in the case of Gaussian rewards the dynamic program is considerably faster. However, the integer programming is still relatively efficient and is more readily generalizable to non-Gaussian distributions.

Kleywegt (1996) introduces the *dynamic and stochastic knapsack problem* (DSKP). In this case, objects of random benefit and weight arrive over time via a stochastic process and are examined for inclusion in a finite knapsack. At the moment of arrival, the benefit and weight become known. Items must be accepted or rejected immediately. Rejected items are lost. Future benefits may be discounted. The objective is to maximize the total benefit, usually at a decision horizon. In its dynamic aspect, this problem is related to optimal stopping problems, a well-known example of which is the *secretary problem* (see Freeman, 1983, for a survey).

In a succession of papers Kleywegt and Papastravou (1996 with Rajagopalan, 1998, 2001) examine different variations of this problem: discrete versus infinite time horizons, fixed weights or benefits, penalties for rejections, discrete vs continuous distributions, and arbitrary probability distributions. The approaches they use draw heavily from dynamic programming and Markov Decision Processes (MDP).

Lu, Chiu, and Cox (1999) look at the project selection problem as a *stochastic knapsack with finite time horizon.* Projects of type k arrive via a stochastic process. For each type k, the weight w_k and reward r_k is known. The object is to maximize revenue by some deadline T. The authors use dynamic programming to explore the case where the horizon T is random.

Van Slyke and Young (2000) describe much the same problem as the *finite horizon stochastic knapsack problem.* The emphasis is slightly different as instead of projects, the arrivals are customers. Again, customers of type k arrive via a stochastic

process, each with weight w_k and reward r_k. The object is to maximize revenue by some deadline. They solve the problem via a continuous time, finite horizon, discrete state, dynamic program. They examine the special case where $w_k = 1$ for all k, which is the airline yield problem.

Motivated by transportation problems, Cohn and Barnhart (1998) step back and revisit the issue of random weights with a known set of objects. In their problem, the objects are customers who must be served but whose demands for service are uncertain. With Gaussian-distributed weights, they create an objective function maximizing reward but with a penalty for the expected violation of the capacity. They describe a branch and bound solution scheme.

Fortz et al (2005) examine the same problem but treat it more formally as a stochastic programming two-stage recourse formulation. The penalty is viewed as an opportunity to buy more resource in the second stage. With the appropriate assumptions, the result is an unconstrained mixed-integer convex non-linear program. For solution, they employ an LP/NLP based branch and bound algorithm (Quesada and Grossmann, 1992) using CPLEX to solve the sub-problems.

Kress et al (2007) develop a combinatoric problem they term the *minmax multi-dimensional knapsack problem.* This is developed as part of a two-stage recourse problem with chance constraints to model a logistical resupply problem, where the second stage involves satisfying realized supply demands. In contrast with most chance constraint problems found in the literature, the second stage random variable is an

arbitrary discrete distribution. The chance constraint requires a combinatoric formulation using the concept of *p-efficient points*.

Brown et all (2007) provide a survey of the methods involved in military capital planning. These addressed the deliberate multi-year planning involved in optimized multi-billion dollar portfolios where the funding options and their associated reward and cost are understood. However, the complex nature of these options are such that an accurate portrayal of the trades involved requires the explicit modeling of fixed and variable benefits and costs, interactions between systems, and the effects of lot-purchases and multi-year planning horizons. Non-linearities are typically approached via piece-wise linear functions. These approaches result in large-scale mixed-integer programs. They discuss means of improving solution times, provide examples, and delve into the issue of measuring the reward in a military context.

Keles and Hartman (2007) describe an approach for optimizing multi-stage research and development (R&D) portfolios, which they consider to be a specialized case of the dynamic stochastic knapsack problem. Their problem statement is very similar to the one given here for the Wartime Portfolio Selection problem but their focus is on the pharmaceutical R&D portfolio. They show how the sequence of decisions to fund (continue funding), delay, or terminate R&D projects within (or entering) a portfolio can be represented as a stochastic dynamic program. Given the size of practical problems, they solve via an approximate dynamic programming approach.

Powell (2007) describes how approximate dynamic programming (ADP) grew as an approach for solving dynamic programs too large to solve via conventional means. In DP, the value function provides the optimal value of the objective function for a given state of the system. A common theme for DP problems is that they can all be thought of as the shortest path in a network, where the nodes of the network are states of the system. While DP is efficient in that it solves for the shortest path without having to enumerate every path, it still requires that every state/node be visited. For complicated problems, where the nodes may exist in an N-dimensional state space, this is computationally infeasible. ADP uses statistical methods to approximate the value function without having to visit every state. We delve more deeply into approximate dynamic programming in chapter 5.

2.3 Applicable Military Value Literature

Profit is traditionally used to measure value in business applications. However, a military organization is not a profit-making enterprise. Military investment demands can range from new weapon systems, the means to move them, the means to target for them, the means to maintain these systems, to the manpower to operate them, to include the facilities to house the manpower's families. None of these lead to a profit. Therefore, measuring military value requires a way to quantify the degree of accomplishment of military goals and objectives.

Military requirements such as firepower, mobility, or targeting capability can be treated as constraints to satisfy while minimizing costs, an approach taken by Dell and

Tarantino (2003). More commonly, the approach is to develop constructed measures of benefit as linear combinations of various normalized measures that seek to capture the level to which multiple, frequently competing objectives are satisfied by bundles of choices. This approach was advocated by Keeney and Raiffa (1976), and in the military context is described by Brown et al (2007), and more extensively by Parnell (2007).

The measures that might be employed are very specific to the strategic goal being examined. Perry (2007) discusses the challenge of mapping the performance of a system to attainment of strategic goals, including measures that may address the improvement in outcome as a result of using different combinations of systems.

Crain (2007) provides an overview to the practice of Theater Campaign Analysis. This framework looks at the entire campaign: mobilization, deployment, employment, redeployment, and sustainment. This framework may be applicable to supporting JIEDDO's objective to acquire the best bundle of counter-IED in support of two active theater campaigns. Applying Theater Campaign Analysis to JIEDDO's problem would require consideration of how counter-IED initiatives contribute to theater campaign goals.

Bertha and Shelton (2007) discuss the topic of Combat Operations Analysis. Broadly, these analyses are more concerned with exact modeling of the combat operations in the employment phase of the campaign. Much of the focus is on the modeling of traditional force-on-force, large-scale combat, which the counter-IEDs campaign is not.

Orgeron (2007) discusses the specific issue of small-scale contingency analysis. The general thrust of his discussion is on the use of this analysis to determine future requirements, versus the model of a specific campaign, and the means to determine cause and effect that may lead to answering questions about the best bundle of capabilities.

A major aspect of the counter-IED problem involves search. The primary function of a number of IED initiatives is to search for IEDs. Washburn (2002) provides a survey of methods to address search problems.

A large number of IEDs occur on roads. Washburn (2007) describes a network interdiction model that incorporates the means to model IED and counter-IED effectiveness. In our context, his approach could prove useful for addressing how to model the cumulative effect of different counter-IED initiatives operating in the same tactical space.

A particularly pernicious problem is the suicide bomber. In contrast with the road-side IED, which is an ambush, the suicide bomber is an attack. Kaplan and Kress (2005) conduct an analysis of the effectiveness of suicide bomber detection methodologies for the protection of urban areas from suicide bomber seeking mass-casualty targets. A military context for the use of these technologies may be more focused on the protection of specific sites.

2.4 Potential Contribution

The multi-stage R&D portfolio optimization problem described by Keles and Hartman (2007) appears most closely aligned with this problem. The main differences lie

in that they are examining a private sector problem of selecting pharmaceutical R&D projects, where the reward can be measured in dollars, the types of projects are homogenous, and the distribution of cost per project is relatively bounded. In the wartime portfolio selection problem, the means of valuation are not so clear and the projects are heterogeneous, which may alter the methodology for optimal selection of the portfolio.

The specific problem of measuring the military value of counter-IED initiatives, which provides a case study for the more generic problem of wartime portfolio selection, may not have broader application beyond the counter-IED problem itself – but the IED is considered by many to be one of the most pressing technological issues facing western armies today. Additionally, while JIEDDO at its height had annual budgets on the order of $3-4B a year, these funding levels have already started to diminish, increasing the difficulty of the resource allocation decision. Enhancing JIEDDO's ability to acquire an effective counter-IED portfolio at a lower level of resourcing would be a valuable contribution.

CHAPTER 3 – MEASURING THE BENEFIT OF WARTIME PORTFOLIOS

3.1 Introduction

The benefit or value of military items is typically non-fungible; it cannot be measured in dollars. While many approaches for measuring preference quantitatively are known in the decision analytic literature, JIEDDO does not currently employ any means to measure the benefit of its initiatives (GAO report 2007). Since JIEDDO represents a powerful case study for the wartime portfolio decision problem, we present our approach for measuring initiative value in the case of this particular portfolio. We conclude the chapter by considering what aspects of the decision analysis approach can be generalized.

3.2 JIEDDO Overview

During the conflicts in Iraq and Afghanistan the most lethal tool of the insurgent has been the Improvised Explosive Device (IED). In response to this threat, Congress in January 2006 established the Joint Improvised Explosive Device Defeat Organization (JIEDDO) with the mission of leading, advocating, and coordinating U. S. defense actions aimed at defeating the IED as a weapon of strategic influence. Given the urgency of wartime, JIEDDO seeks to deliver solutions to meet counter-IED needs of U.S. forces in the areas of conflict as quickly as possible.

Counter-IED solutions cover a broad range of both military functions and forms. Functions include intelligence, surveillance, electronic warfare, maneuver, targeting, fire support, force protection, and information operations. Forms include software, airborne sensors, ground sensors, vehicle systems, armor recipes, jammers, spoofers, scanners, robots, and a gamut of contracted services.

To help manage these efforts, JIEDDO partitions the counter-IED solution space into three *Lines of Operation*: Attack the Network (AtN), Defeat the Device (DtD), and Train the Force (TtF).

AtN is focused on preventing IEDs from reaching the intended place and time of employment on the battlefield. AtN targets insurgent activities to include financial, recruiting, training, logistical, manufacturing, planning, command and control (C2), and operational functions.

DtD focuses on solutions that defeat the IED once it has reached its intended place of employment. DtD attempts to detect, neutralize, or mitigate all aspects of the IED itself, to include its triggering systems, its arming systems, its firing systems, its means of concealment, its means of delivery, and its means of lethality.

TtF addresses the means to gain and maintain force readiness for the counter-IED fight, primarily concerning the gaps in the Services' ability to prepare their forces for the latest IED threats and counters.

JIEDDO ties this all together via the Joint IED Defeat Capabilities Approval and Acquisition Process (JCAAMP). JCAAMP is designed to rapidly usher promising

solutions through a series of phases culminating in transition of the successful solutions to one or more of the Armed Services. JCAAMP's goal is to compress standard defense procedures in order to deliver solutions to the warfighter field in months instead of years. To maximize its responsiveness to the warfighter's needs, JIEDDO considers initiatives sequentially as they arrive. Because the Services are the ultimate customers in this process, the Services, the Joint Chiefs of Staff, and the Office of the Secretary of Defense are active participants in all JCAAMP decision points.

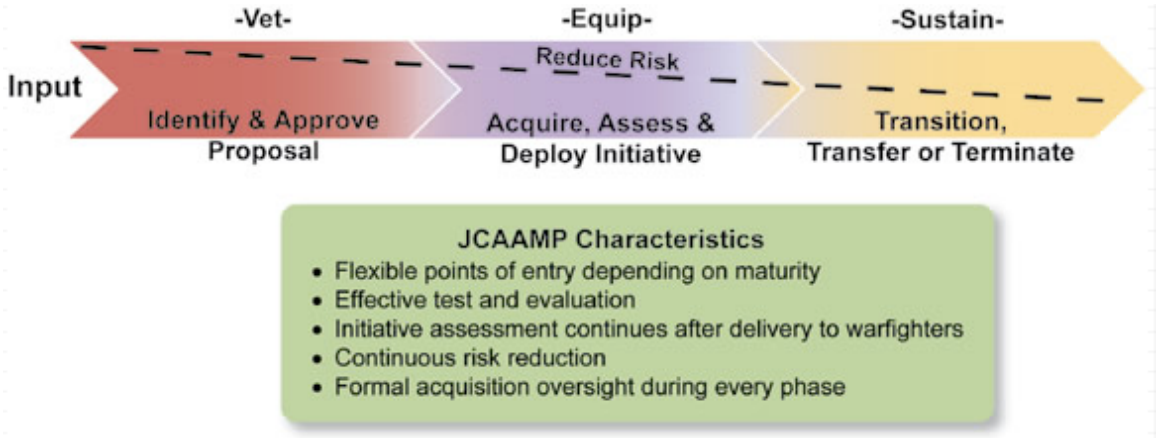

Figure 3-1 The Joint IED Defeat Capability Approval and Acquisition Process (JCAAMP)

As shown in Figure 3-1, JIEDDO continuously seeks proposed solutions and rapidly vets those it finds. A vetted solution of sufficient maturity and suitability becomes an "initiative". After an accelerated period of development and testing, a

suitable quantity of the initiative is acquired and deployed to theater for use by select units in order to assess the initiative's combat effectiveness.

If an initiative has been successfully demonstrated in combat as a counter-IED solution, JIEDDO has two years during the sustainment phase to arrange for the solution's turnover to one of the Services. This time period coincides with the length of the defense budgeting cycle, allowing the Services the time required to budget for the assumption of ownership of a new solution. Every effort is made to identify the likely transition Service as early as possible.

Between FY06 and FY10, JIEDDO received $17.4B in appropriations (JIEDDO, 2010). Congress has been concerned about the efficacy of JIEDDO in employing these funds and has directed several assessments of JIEDDO by the GAO, to examine, among other things, JIEDDO's management practices, performance measures, and metrics. GAO's studies have provided a variety of recommendations regarding the development of metrics for the selection of initiatives and tracking the performance of initiatives (GAO, 2008)

3.3 Military Value Literature

Profit is the standard measure of value in business applications. However, a military organization is not a profit-making enterprise. Military investment opportunities include new weapon systems, training systems and facilities, transportation systems, manpower, healthcare, and the facilities to house manpower and their families. None of

these led to profit. Measuring military value requires a way to quantify the degree of accomplishment of military goals and objectives and aggregate to a common scale.

Military requirements such as firepower, mobility, or targeting capability can be treated as constraints to satisfy while minimizing costs, an approach taken by Dell and Tarantino (2003). However, when the problem is to maximize military capability subject to constraint, a method is needed to measure this capability that encompasses the many ways capabilities may be manifested: intelligence, logistics, fire support, maneuver, to name a few.

A common approach for assessing the strategic value of investment decisions is multi-objective decision analysis (MODA) (Keeney and Raiffa, 1976). MODA is a valuable technique for complex problems with multiple stakeholders, complex value trade-offs, significant outcomes, and major uncertainties. Most MODA applications consist of a hierarchy of goals and objectives, evaluative measures aligned with the goals and objectives, value functions to translate evaluative measure levels to a common scale, and weights.

A general description of the use of MODA for military budget optimization in a mathematical programming context is described by Brown, Dell, Loerch, and Newman (2007). Several descriptions of specific MODA applications in a military budgetary context are readily available.

Loerch, Koury, and Maxwell (1999) describe how they employed value-added analysis to optimize U.S. Army long-range budgets by blending: large-scale campaign

simulations of multiple scenarios; experimental design to isolate system contributions; multi-objective decision analysis to measure the value of a given budget; and a mixed-integer program to identify optimal budgetary solutions.

Parnell et al (2002) employed future value analysis to support National Reconnaissance Office resource allocation. This approach employed structured interviews to identify futures challenges and opportunities, a multi-objective decision analysis using value-focused thinking, and integer programming for optimal resource allocation.

Parnell et al (2003) employed a similar approach to help identify the optimal portfolio of R&D solutions across a diverse set of capability areas but with a significant difference. As before, the portfolio value model employed a linear combination of capability area scores. The difference lies in how the capability scores were obtained. Instead of a capability score that resulted from an additive function of selected solutions, the score was the maximum from amongst the selected solutions within a specific domain. Solutions that did not offer a materiel improvement in capability over the current portfolio did not contribute value.

Parnell (2007) describes various practical approaches to the application of MODA with regard to determining values. These include "gold," "platinum," and "silver" approaches. In the gold standard approach, the values in the MODA model are built on an approved vision, strategy, policy, or other high-level guidance. The platinum approach is based on direct interaction with the decision-maker, while the silver approach

is based upon input obtained from the decision-makers representatives. These approaches are often used in combination, and the resulting model must be presented to decision makers for refinement and validation. In our research, we have followed a combined approach, employing reviews of JIEDDO's strategy [JIEDDO, 2009], interviews with intermediate level personnel, and careful observation of JCAAMP functions over a period of a year.

3.4 Strategic Goals and Considerations

A wartime program must have as its principle aim the most effective collection of solutions within its capability area relative to the war effort. JIEDDO's mission is to enable the defeat of the IED as a weapon of strategic influence. JIEDDO is not a military service and thus does not actually fight. Rather, it energetically seeks to aid the Services and theater commanders by gaining for them the means to reduce the effects of the IED.

A key criterion for selecting a solution is its potential to contribute to the counter-IED campaign. Thus, our value model first seeks to measure the initiative's ability to contribute to these efforts. From our research we have identified two additional goals.

JIEDDO wants initiatives that can be developed and deployed to the theater of war as quickly as possible. A perfect approach that is not ready until after the war is concluded is of little use.

JIEDDO also wants initiatives that are likely to transition to one of the military services. By design, JIEDDO only funds initiatives for a fixed period of time, which is

intended to be sufficient time for the initiative to prove its usefulness. Initiatives should then either transition to a service's ownership or they are terminated. Because of the long lead time required for the services to identify means of funding an initiative should it transition, JIEDDO pushes to identify the likely service to gain ownership of the initiative early on in the initiative's life cycle. From our observation, many factors appear influence this decision. These include the maturity of solution, the total ownership cost, and the "endurance" of a solution. An initiative that is easily countered may have immediate impact but no long-term value. However, a strong short-term impact may still be worth the investment.

We describe a decision analytic methodology for quantifying counter-IED value. It has three components: a MODA portfolio model to assess the value of a counter-IED portfolio and thereby the potential net portfolio value of an initiative; a model for eliciting probabilities that an initiative will transition; and model for assessing how much to discount initiative value over the expected time for the initiative to become operational.

3.4.1 Qualitatively Value Model

Counter-IED Value Hierarchy. The MODA portfolio value model aligns with JIEDDO's lines of operation: AtN, DtD, and TtF. Through our interviews of personnel, review of documentation, and a year's worth of observation, we developed a multi-

objective decision model of Portfolio Counter-IED Value (PCV). Figure 3-2 depicts the PCV goal hierarchy.

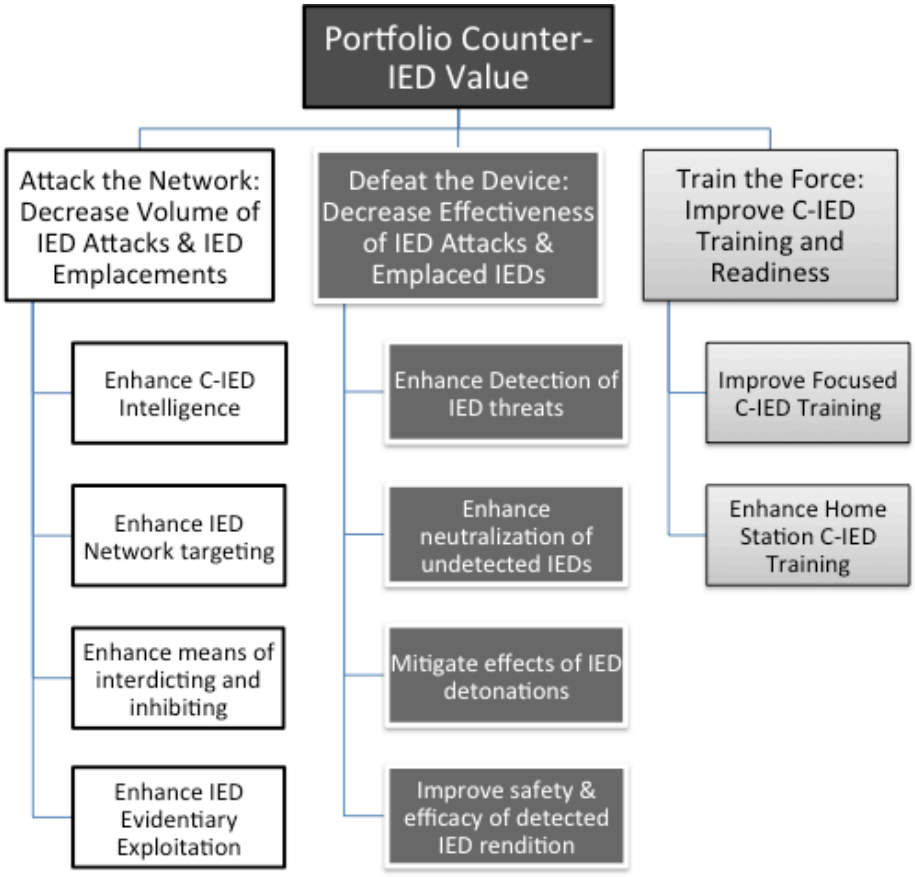

Figure 3-2 Proposed value hierarchy for measuring portfolio counter-IED value

Each top-level (Tier 1) goal corresponds to a line of operation. AtN has as its primary objective to reduce the volume of IED emplacements and attacks. This function focuses on preventing IEDs from reaching the enemy's time and place of employment.

DtD has as its objective to decrease the effectiveness of emplaced IEDs and attacks. These IEDs have reached the enemy's intended place and time of employment and the desire is to detect them and render them safe for exploitation; to, barring detection, somehow neutralize them: or should detection and neutralization fail, to mitigate their effects. Lastly TtF seeks to improve C-IED training and readiness.

Attack the Network Value. We model the AtN aspect of the campaign as a cyclical concept with four domains: counter-IED intelligence, IED network targeting, interdiction/inhibition of attackers, and IED evidence exploitation. As depicted below, these domains come from a conceptual model we developed to depict how AtN functions relative to the enemy. Generally AtN initiatives will contribute to only one of these domains. These domains form the Tier 2 goals under the Tier 1 AtN goal.

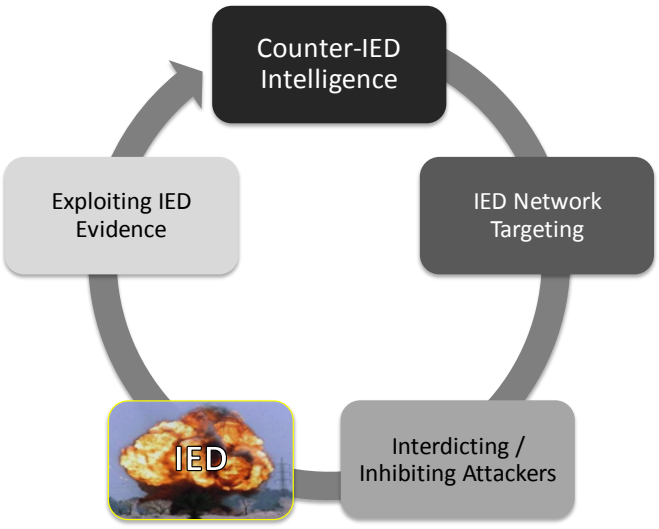

Figure 3-3 The Four Domains of the Attack the Network Cyclical Model

The first aspect of attacking the network is to maximize the intelligence available to all those activities involved in the counter-IED fight, from stateside agencies to infantry squads planning their next patrol. The next domain is more specialized and offensive in nature: how to improve warfighter's means to target IED cells and functions.

IED cells that escape being targeted might still be degraded by the allied forces denying or inhibiting their freedom of movement. The interdiction/inhibition domain covers this conceptual "battle hand-off" between AtN and DtD. It specifically aims to interdict or inhibit the IED in its final leg towards its intended destination. The last domain is the technical ability of the Force to exploit the information gathered from each encounter with the IED, which includes forensic methods. This domain completes a natural feedback loop to the first step of enhance C-IED Intelligence.

Defeat the Device Value. DtD Tier 2 goals stem from a conceptual event tree, shown in Figure 3-4, that depicts a defense-in-depth. The first line of defense is to successfully detect the IED before the friendly entity (commonly a vehicle) and the IED come within the IED's effective range. If the IED is not detected, the next line of defense lies in technologies that can neutralize the device: prevent it temporarily or permanently from operating as intended. If neither detected nor neutralized, the IED may detonate as intended. The last line of defense is a system's capability to mitigate the IED's lethal effects.

If, however, the IED is successfully detected, there remains the sticky issue of how to safely, rapidly, and effectively clear and make safe detected IEDs, particularly since they are valuable sources of intelligence. This goal links to the AtN Tier-2 goal of Exploiting IED Evidence. While all IED events can be investigated, the unexploded IEDs are better sources of intelligence than the detonated ones.

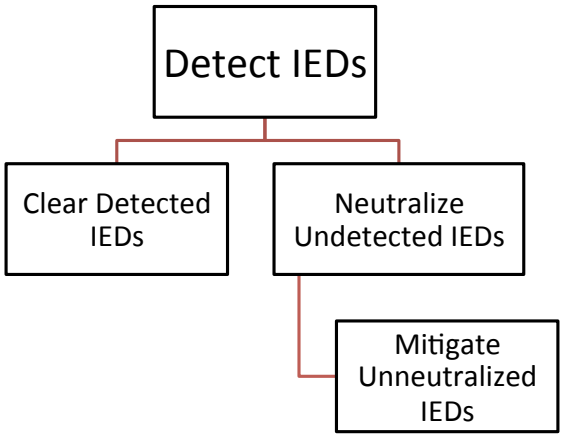

Figure 3-4 DtD Event Tree

Train the Force Value. TtF is subdivided into two basic training needs that correspond to the Tier 2 goals. Focused IED Training provides for specialized training events, courses, and facilities focused on counter-IED (National Training Center, schools, etc). Home Station Training seeks to provide units with enhanced means to conduct their

own counter-IED training at their home stations prior to deploying to theater. In either case, the desire is to maximize readiness of individuals and units by ensuring they have the best training with the latest information from theater.

3.4.2 Quantifying Value

Overview. In keeping with Parnell et al (2003), we seek to measure first the capability of the portfolio of solutions, and then measure of an initiative's value as its marginal contribution to the portfolio. The MODA model we describe uses the weighted sum of portfolio capability level across the Tier 2 goals.

Evaluative Measures. In decision analysis, each goal requires a means of measuring goal realization. The Tier 2 goals shown exist at relatively high levels of aggregation. Developing the evaluative measures for each of these goals requires follow-on research and is beyond the scope of this dissertation. For the purposes of our analysis, we assume that the means of measuring goal fulfillment exist for each Tier 2 goal, and will refer to these means as Tier 2 *evaluative measures*, indexed by *m*. The portfolio's capability level attained on each measure *m* is designated y_m and the vector of these measures is $\boldsymbol{y}$.

Value Functions. As described by Parnell (2007), each Tier 2 evaluative measure *m* needs a least preferred level and a most preferred capability level. The least preferred level, also called the threshold level, is the level below which no value is contributed. The most preferred capability level, or objective level, is the level above

which no further value is accrued. Between these endpoints, i.e. in the range of variation, the capability level on the measure, y_m, requires a single dimensional value function, $v_m(y_m)$, which maps from the measure space to the normalized value space; e.g., a point on the interval [0, 1], [0, 1000], or [0%, 100%]. These value functions should be elicited from decision makers' preferences, which could include using the threshold and objective levels from requirements documents.

Swing Weights. We should not expect that the maximum possible contribution of each value function $v_m()$ to be equal. If they were, we need only take the average of the values. Accordingly, we need to assign to each $v_m()$ a weight w_m. Parnell describes these as swing weights, which reflect not just importance but the change in overall value within an evaluative measure when swung from its least preferred level to its most preferred level. He describes several approaches for eliciting these from decision makers.

Portfolio Counter-IED Value (PCV). We obtain the resulting mathematical model for measuring PCV, which is the weighted average of normalized value functions of the capability levels.

Equation 3-1: Portfolio Counter-IED Value

$$PCV(\boldsymbol{y}) = \sum_m w_m \, v_m(y_m)$$

$$\sum_m w_m = 1, w_m \geq 0$$

3.4.3 Valuing Individual Initiatives

For portfolios of financial instruments, Markowitz (1959) developed a mathematical programming approach that constructed the portfolio by considering the expected return of the entire portfolio and the covariance of its components. There are a couple of insights to gain from this approach. The first is to understand that initiatives may interact with other solutions in the portfolio, thus inherently changing the value of the portfolio in potentially non-linear ways. The second is that the value of an initiative must be based on its net contribution to the total value of the portfolio: the difference between the value of the portfolio *with* the initiative and the value of the portfolio *without* the initiative.

Loerch, Koury, and Maxwell (1999) modeled the value of all the U.S. Army's Major Defense Acquisition Programs by looking at their contribution to the overall portfolio value of U.S. Army's projected inventory via design of experiments and combat simulations. To consider the independence issue, they also considered combinations of candidate systems. Given the large number of systems, they limited themselves to a factorial design with 2-way combinations.

Parnell et al (2003) were examining space systems and did not have computational models for considering overall portfolio value. Their approach was to score initiatives on each evaluative measure as the maximum of either the initiative's stand-alone score or its score when in concert with the current capability of the portfolio.

Vector Union Operation. To implement this procedure we define a vector union operation $\boldsymbol{x} \cup \boldsymbol{y} = \boldsymbol{z}$ where $z_m = \max(x_m, y_m)$ for all m. This operation ensures that an initiative only is considered if it provides capability higher than the current portfolio in at least one evaluative measure.

To illustrate, if $\boldsymbol{y}$ is the vector of the portfolio's capability level then let $\boldsymbol{x}$ be the vector of the capability levels attributable directly to a new initiative i. Consider the case where only one measure m is affected. Let the new capability level on m be x_m, while the original portfolio capability level on m was y_m. The new initiative scores a 0 on all measures but m and we assume that $x_m > y_m$. The procedure is illustrated below.

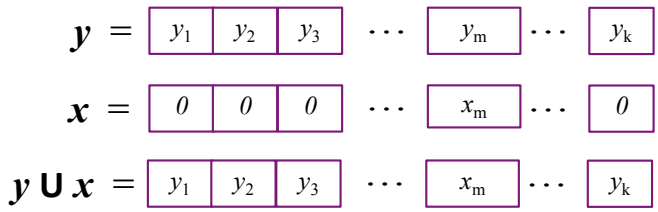

Figure 3-5 Illustrating the vector union operation where xm > ym

These illustrations show only one measure changing at time because we assume a single function initiative. Obviously, multi-mission systems may alter more than one measure capability level at a time.

The Net Initiative Value of i ($NIV(i)$) is the net change in PCV from introducing the initiative i.

Equation 3-2: Net Initiative Value

$$NIV(i) = PCV(\boldsymbol{y} \cup \boldsymbol{x}) - PCV(\boldsymbol{y})$$

Interactions. An issue to consider is the effect of multiple initiatives arriving close together in time. Interactions between these items might be missed by this process. One approach might be to ignore these interactions, since in this sequential process only a few initiatives are ever considered at the same time, and it is unlikely that any two will interact.

A more rigorous approach is to require that the evaluation process consider potentially interacting initiatives that arrive simultaneously both separately and in combination. This complicates the issue, adding potentially many more candidates solutions to evaluate – for any set of arrivals with cardinality *n* where ALL the arrivals might possibly interact, then $2^n - 1$ solutions would need to be evaluated. However, the trend we observed at JIEDDO was that only a few items were ever evaluated simultaneously at any single transition point, and the potential for interactions proved rare.

For example, if two initiatives, A and B, arrive in the same week. Both are assessed to provide a modest improvement over the current portfolio on their own. However, the assessment reveals the two may interact synergistically – their contribution in combination is greater than the sum of their individual contributions. In this case, there are four mutually exclusive options for the leadership: buy nothing, buy A and not B, not A and B, or both A and B.

3.4.4 Assessing the Likelihood of Transition

The ideal goal is for funded initiatives to prove so effective that they eventually transition into the permanent inventory. An example of this is the now ubiquitous Counter-IED Radio-controlled Electronic Warfare (CREW) jamming technology, the development of which was funded by JIEDDO (HASC Report 2008).

Not all solutions transition into the permanent inventory. Some prove ineffective when tested in realistic environments, to include limited evaluations on the battlefield. Others do prove effective but are simply too expensive for the benefit they offer. Furthermore, because the process of securing resources in the Planning, Programming, Budgeting and Execution System typically takes years, the process of transitioning an initiative has to start very early in its life-cycle to ensure that, should the initiative prove to be successful, the receiving Component will have the required resources to support the initiative.

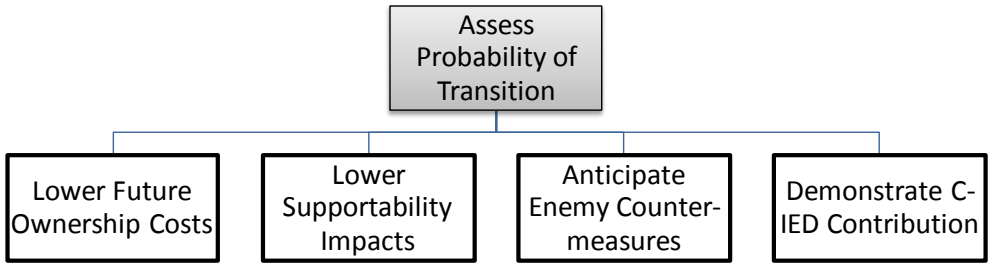

Figure 3-6 Factors for Assessing Likelihood of Transition

To aid in in the selection process, we propose that the probability of transition should be systematically assessed, independent of the PCV process. Based on our observations at JIEDDO, the main reasons services might be disinclined to support transition of an initiative, independent of its potential counter-IED effectiveness, include the following: an initiative was too expensive, it had onerous support requirements, it was easy to defeat in the long term, or it lacked demonstrated evidence to confirm the potential value. We envision using an assessment of these factors to elicit from decision makers an estimate of the likelihood of transition (see Figure. 5). We use $P_T(i)$ as the probability of transition of initiative *i.*

3.4.5 Accounting for the Value of Time

Meeting the demands of the warfighter requires that solutions be fielded as rapidly as possible. However, developing new solutions takes time. Discounting is a means of estimating the value today of something not expected to yield value until sometime into the future. In finance, usually a benchmark is used, such as a current or projecting lending rate, which may or may not be adjusted for inflation. This benchmark rate is then raised exponentially based on how long it will take to realize the expected value. This is the basis for the Net Present Value calculation.

A military solution available now is clearly preferable to one not available for some time. When comparing the value of two military solutions of equal estimated effectiveness, the value of the one not immediately available should be discounted. We

argue that all else being equal, the discount factor is a measure of the willingness of the warfighter to wait for a solution. Some factors that might influence the warfighter's willingness to wait include whether a solution fulfills a validated but unfilled urgent requirement (e.g, a specified Joint Urgent Operational Need (JUON)). A step down in this willingness to wait might be if a solution did not have an unfilled urgent requirement but did fit within an anticipated future need found within a document such as a technology roadmap plan. Lastly, the warfighter might discount steepest for a solution that did not have any associated statement of need but perhaps provides an unanticipated capability. Table 3-1 illustrates a posited three level base discounting scheme.

Table 3-1 Posited Scheme for Determining Base Discounting Levels

Level	*DF* (posited)	Rationale
I	0.99	Has validated war-fighting requirement (e.g., JUONS)
II	0.95	Solution fulfills an anticipated need (e.g., technology road map)
III	0.90	Solution has no associated statement of need

Some solutions in the bottom level could prove to be unanticipated, ground-breaking solutions. Should these be immediately available, then there is no discounting. If there is a long wait before these might be ready, then discounting would affect these heavily. Of course, over time the existence of some potential solutions may influence the development of requirements leading to a change in their discounting level.

3.4.6 Discounted Expected Net Initiative Value

We demonstrate how NIV, likelihood of transition, and discounting come together. Upon arriving before a decision-making body, an initiative is assessed for its potential to provide an enduring contribution to the counter-IED fight. This is done via the PCV model, where the value of the portfolio is measured with the initiative, and the resulting increase in value over the current portfolio is assigned to the initiative as its NIV.

To become an enduring contribution, which we assume to be coincident with realization of NIV, the initiative must first transition to a Service. Since NIV is not certain, we should employ some method of adjust for this uncertainty. At this stage in the model's development, we are considering only two possible outcomes: transition and termination. Treating this as a simple Bernoulli variable, we take the expectation of the NIV, using the probability of transition P_T.

Taking only the expectation assumes that the time until NIV is achieved is insignificant. The realities of conflict dictate a strong preference by the warfighter for solutions that work sooner rather than later. To account for the cost of waiting we employ a discount factor $DF^{t(i)}$, where t is the delay until solution i can be deployed to the theater of conflict, and where DF the discount factor rate describes the warfighter's willingness to wait for the solution to arrive on the battlefield. A likely unit of time to use when discounting is the fiscal quarter.

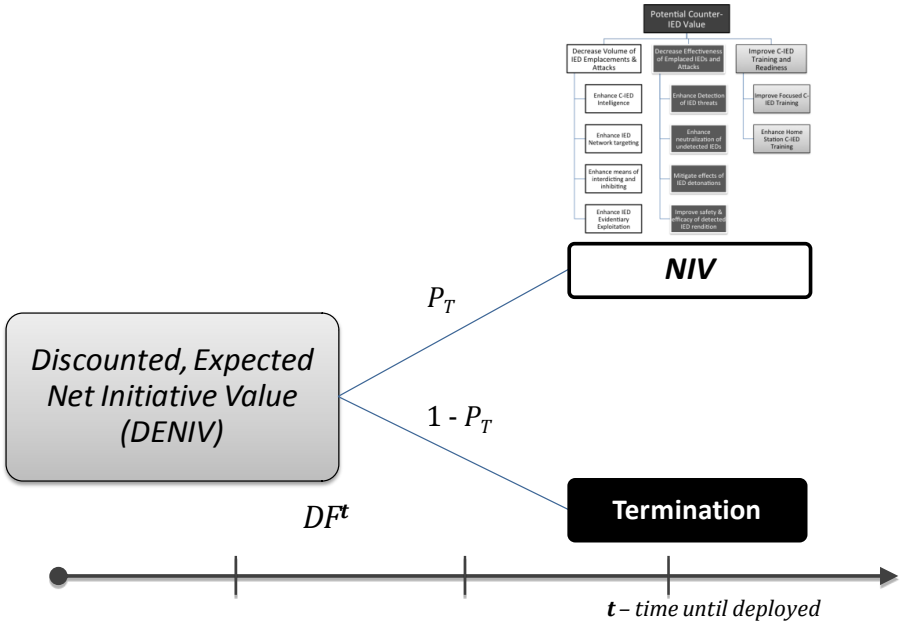

Figure 3-7 Discounted Expected Net Initiative Value

As the equation below shows, *DENIV*(*i*) is the discounted, expected, weighted average of its net contribution over the present portfolio.

Equation 3-3: Discounted Expected Net Initiative Value

$$DENIV(i) = P_T(i)DF^{t(i)}[\left(\sum_m w_m\, v_m(\max\,(y_m, x_m))\right) - (\sum_m w_m\, v_m(y_m))]$$

This approach we have described provides a means to quantify value, but it does not necessarily describe how to optimize value. That is subject of our ongoing research. However, reader's familiar with dynamic program may recognize strong similarities between Equation 3-3 and some forms of Bellman's equation.

3.5 Numerical Example

We provide a numerical example to illustrate how DENIV might be employed as part of JCAAMP or a similar process. Let us assume that on a periodic basis, the counter-IED development agency evaluates the collective counter-IED capability levels across the lines of operation, and reassesses the vector of swing weights $\boldsymbol{w}$.

Based on this, the agency's leadership assessed the swing weights to assign to each of the enterprise level attributes. These weights reflect not just the importance of each attribute but the amount of swing available between the current level and the desired ideal. The overall portfolio value is the weighted average of the capability levels.

Table 3-2 Swing Weights – Illustrative example

Goal	Current Capability Level	Rank	Raw Swing Wt	Normalized Swing Wt
C-IED Intel	50%	3	45	0.129
IED Cell Targeting	20%	1	60	0.172
Interdict & Inhibit	20%	2	50	0.143
Evidence Exploitation	70%	7	30	0.086
IED Detection	50%	5	40	0.115
IED Neutralization	20%	4	41	0.117
IED Mitigation	30%	6	35	0.100
IED Clearance	80%	10	10	0.029
Focused Training	70%	9	15	0.043
Home Station Training	50%	8	23	0.066
Unweighted Average Capability Level	46.0%	Weighted Average Capability Level		38.5%

As the table shows, the swing weights result in a overall capability level different than the unweighted average – in this particular case lower. This will result in a higher priority being given to solutions in the higher weighted areas.. In the charts below the capability level is the area under the bars. In the unweighted scheme the bars have the same width, while in the other the widths of the bars reflect the respective swing weights.

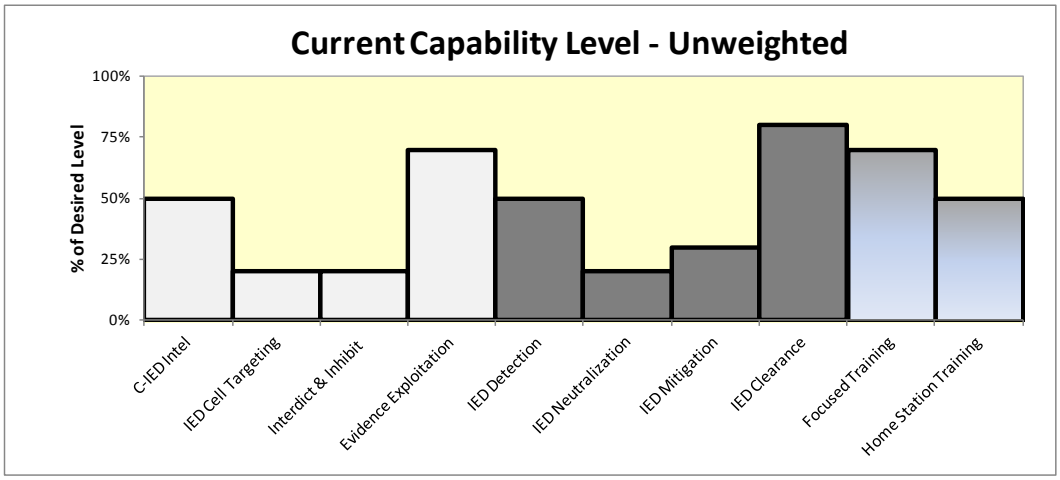

Figure 3-8 Unweighted Assessed Capability Levels vs. Objective Levels

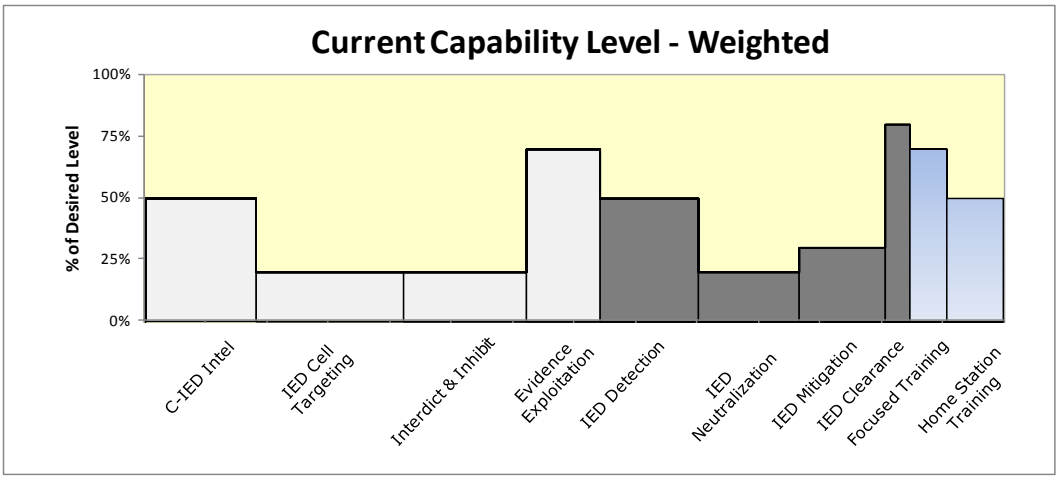

Figure 3-9 Weighted Assessed Capability Levels

The following week, three initiatives arrive: Ground Sensor A, Intelligence Analyst Software B, and Training System C. The agency's solution evaluation board (SEB) has convened to examine these.

As briefed, Ground Sensor A detects a particularly lethal and elusive class of IEDs 60% of the time. This is a three-fold improvement in U.S. forces current probability of detection. This class of IEDs causes 40% of all IED casualties. Thus, in terms of coalition forces' total ability to detect all types of IEDs, as weighted by the casualties these IED's types cause, Ground Sensor A represents a 20% improvement. The SEB is informed that this system has been successfully employed in recent contingency operations by an ally and requires minimal levels of sustainment support. The Service reps find that its overall costs are affordable. Thus, its probability of transition is set at a high level – 0.9. Ground Sensor A is addressed by a JUON and is

thus assigned a DF of 0.99 - the highest discount rate employed by the SEB. It can reach theater in the next fiscal quarter, so $t = 1$ and resulting discount factor is $0.99^1 = 0.99$

The SEB is informed that the Intelligence Analyst Software B significantly increases the productivity of a large swath of intelligence analysts. It is estimated that it will enhance overall C-IED intelligence by 33.3%. However, as it has not yet fully matured, it is expected to have high sustainment costs, particular in the forecasted number of developers and help desk staff requirements. Thus, the SEB assesses its probability of transition at 0.7, indicating some concern for this program. Additionally, it is not forecasted to be operational for another 9 months and has no JUON or other supporting requirement. By the business rules used in our fictional example, a lower DF of 0.9 is used. Since the operational availability is not forecasted to be for another 3 quarters, the resulting discounting factor is $0.9^3 = 0.729$.

Training System C provides a 10% improvement in home station training throughput and cuts in half the lag time in inserting the latest battlefield lessons-learned into the training: a 50% improvement in overall home station training metric. It has been demonstrated at one National Guard site, but the Army and Marine Corps will need JIEDDO to help fund a full roll out of the system, which will take about 12 months. There is much concern about recent environmental issues with its employment that may or may not be resolvable. Despite its modest sustainment requirements, this last issue caused the board to assign this initiative a 0.5 probability of transition. A like capability

is addressed in the TtF technology road map so its DF is an intermediate value of 0.95. The forecasted year long wait results in a discount factor of $0.95^4 = 0.814$.

The results from the board's evaluation are shown in the table below. Ground Sensor A scored highest, but in large part because its maturity, high likelihood of transition and readiness to be deployed. In fact, the act of computing the discounted expectation reversed the rank ordering of the initiatives that resulted from measuring the just their weighted overall value. This highlights that resolving the key issues with Systems B and C – perhaps getting the theater commander to provide a JUON for System B and resolving the environmental issues with System C – could have dramatic effects on their scores.

Table 3-3 Numerical Example DENIV Results

Evaluative Measure	Current Capability Level	With System A	With System B	With System C
C-IED Intelligence	50%	50%	**67%**	50%
IED Cell Targeting	20%	20%	20%	20%
Interdict & Inhibit	20%	20%	20%	20%
Evidentiary Exploitation	70%	70%	70%	70%
IED Detection	50%	**63%**	50%	50%
IED Neutralization	20%	20%	20%	20%
IED Effect Mitigation	30%	30%	30%	30%
IED Reduction	80%	80%	80%	80%
Focused Training	70%	70%	70%	70%
Home Station Training	50%	50%	50%	**75%**
Weighted Overall Value	**38.5%**	**39.9%**	**40.6%**	**40.1%**
NIV(i)	**na**	**1.4%**	**2.1%**	**1.6%**
***t* - Time to Deploy**	**na**	**1**	**3**	**4**
DF	**na**	**0.99**	**0.90**	**0.95**
DF^t	**na**	**0.99**	**0.73**	**0.81**
$P_T(i)$	**na**	**0.90**	**0.70**	**0.50**
DENIV(i)	**na**	**1.3%**	**1.1%**	**0.7%**

As the systems progress through the development process, the intent would be to update these measures as information becomes more current while addressing shortfalls that could increase the likelihood of transition or mitigate the effect of discounting. Note that in this example we do not make a recommendation as to which system to choose. The decision makers could choose all three. The purpose is to illustrate quantitatively how these disparate capabilities could be compared on a single value system.

3.6 Summary and Generalizing to the Wartime Portfolio Problem

The model as specified presents a useful starting point for a counter-IED capability developer or resource provider to develop a decision-analytic process for objectively valuing and conducting trades in their portfolio.

In general, the model also provides some useful considerations for agencies conducting rapid, wartime procurement: a value model to address future or potential value of solutions the measures value based on gaps in the capability set, discounting to address the value of time, and considering the probability that the item will successful join the inventory.

There is certainly room in this approach to incorporate Utility Theory, where one needs to consider the risk attitudes of decision makers. We view this as "next spiral" in the decision analytic model; one to introduce once the base model had gained traction within a using organization.

CHAPTER 4 – SOLUTION METHODS: DYNAMIC PROGRAMMING

The next two chapters address methods for making optimal choices for the Wartime Portfolio Selection Problem (WPSP). This chapter focuses on Dynamic Programming and its derivative, Approximate Dynamic Programming, to solve versions of the WPSP. We start with DP since this was the essential approach that Kleywegt (1996) employed when he introduced the *Dynamic and Stochastic Knapsack Problem* (DSKP), and its variants.

4.1 Introduction to Dynamic Programming

The sequential nature of WPSP lends itself to a dynamic programming approach. Dynamic Programming (DP) largely came about to address the challenge of sequential decision making (Bellman, 1957).

The basic principle of DP is to decompose a large, hard problem into small, easily solved problems and then assemble these sub-problems together so that they solve the original problem. In a simplified case, DP can be thought of as a means to find the shortest path in an acyclic directed network with real arc length. The literature in this field is extensive – see Ajuha, Magnanti, and Orlin (1993) for a complete treatment of this subject. The primary goal of this section is to use a simple case of the shortest-path problem to provide an introduction to the principles of DP.

A network is a set of nodes with arcs linking the nodes, where the arcs each have a distance that must be traveled in order to move from one node to the next. *Directed* means the arcs can only be traversed in one direction, with the direction indicated by an arrow. A path is a feasible sequence of arcs that provides means to travel from one node to another node elsewhere in the network. *Acyclic* means that there if you were to start at any given node and traverse the network, there would be no way to return to the node from which you started.

We can describe this network mathematically as follows. Let $N = \{i\}$ be the set of nodes and $A = \{(i,j), i<j, i,j \text{ in } N\}$ be the set of arcs, with $d(i,j)$ the length of the arcs. Note that instead of giving the arcs their own labels, we use the ordered pair of origin and terminus nodes (i, j) to describe each arc. To facilitate understanding, the definition of the set A specifies that the nodes have been labeled in topological order from the source node 1 to sink node n, where $n = |N|$, so that the label on every node i is always greater than the labels on the nodes in its backwards arc (the arcs that lead to i) and less than the labels on the nodes in its forward arc (the set of arcs that lead away from i). The goal of DP is to find the shortest path in this network from every node to the sink. To illustrate we use a small example of an acyclic directed network.

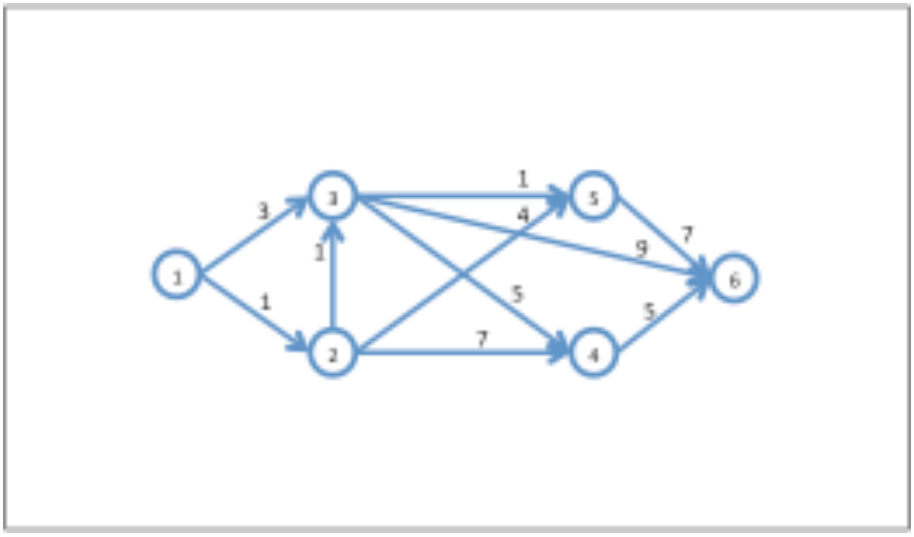

Figure 4-1 Example Acyclic Directed Network

This network starts with node 1. From this source node, any path we choose will eventually lead to the sink node - node 6. The label on each node is less than nodes in its forward arc; e.g., node 2 has nodes 4, and 5 in its forward arc, node 3 has nodes 4, 5, and 6 in its forward arc and so on. Each arc has a length, shown by the number next to each arc. A path is a just sequence of connected nodes along which we can travel in this network. For example, 1-2 is the only path from node 1 to node 2, but there are two paths from node 1 to node 3: 1-3 and 1-2-3.. The length of a path is sum of its arc lengths. Path 1-2 has only one arc of length 1 so the length of the path is 1. Similarly, path 1-3 has length 3. Path 1-2-3 has two arcs of length 1 each, so length of the path is 2. Since we are interested in finding shortest paths, we compare the two paths and find by inspection that the shortest path from 1 to 3 is 1-2-3.

Finding the shortest path from 1 to 3 is illustrative of the principle of DP. The shortest path from 1 to 6 in this small network is not immediately obvious. At first blush, one way to ensure we found the correct answer would be to enumerate all the paths, find their corresponding lengths and determine the shortest one. Enumeration seems daunting, both because there does not seem to be a straightforward way to doing that and because it may take a long time before we are done. But perhaps if we find a structured way to break this down into small, easy problems – problems like finding the shortest path from to 1 to 3 – we might be able to use that structure to assemble these small solutions into the whole solution.

In DP the basic approach is to start at the sink n and work backwards, or *recursively*, determining at each node i the shortest distance from it to the sink. Bellman defined $f(i)$ as the shortest distance from node i to the sink n. While working backward from sink, at each node i we visit, we look at the nodes j in the forward arc of i – the nodes we can visit in one step from i - to see which previously visited node j offers the shortest path from i to the sink. Each of these nodes i represents a choice, one that combines the length of the arc from i to it, $d(i,j)$, with the previously determined shortest distance from j to the sink n, $f(j)$. Mathematically we express this choice problem:

Equation 4-1: Prototype Bellman's Equation

$$f(i) = min_{j \, \varepsilon \, FwdArc(i)} \{d(i,j) + f(j)\}$$

Denardo (2003) calls this the prototype formula of the fundamental equation in DP, *Bellman's Equation.* It is the basic form upon which later versions expand. The structure it is exploiting lies in that at each node we do not have to explicitly consider all possible paths from i to the sink in the acyclic directed network. We need only consider the set of shortest paths defined by the forward arc of i and the labels $f(j)$. We describe the basic algorithm.

Simple Recursive DP Algorithm

Step 1. Given $N = \{i: i = 1, 2,\ldots,n\}$, $A = \{(i, j), i < j\}$

ShortestPathTreeList = empty

Let $f(n) = 0$ # distance from sink to itself is zero

Step 2. For i in n-1, n-2,…,1 # decrementing in order until we get to source

For j in Forward Arc(i)

$f(i) = \min(f(i), d(i, j) + f(j))$

record minimizer j^*

Add (i, j^*) to ShortestPathTreeList

Step 3. Return $f()$ and ShortestPathTreeList

Readers familiar with Ahuja, Magnanti, and Orlin (1993) may be conditioned to the shortest path problem as finding the shortest path from the source to every other node.

As we have seen, in DP the problem is defined as the shortest path from every other node to the sink. The difference is one of perspective. In a sequential decision problem the system is traversing towards a desired destination state. Thus, for any node the system may occupy, we want to know the best path to take from there to the desired destination.

Returning to our small example we will apply our DP algorithm to find the shortest path from source 1 to the sink 6. To begin, we set the distance from the sink to itself as 0: $f(6) = 0$. Working backwards, we go to node 5 and evaluate Equation 4-1 for this node. Node 6 is the only node in the forward arc of 5. Since $d(i,j)$ equals 7 and we already know $f(6) = 0$, per Bellman's Equation, $f(5) = 7$. Similarly, $f(4) = 5$.

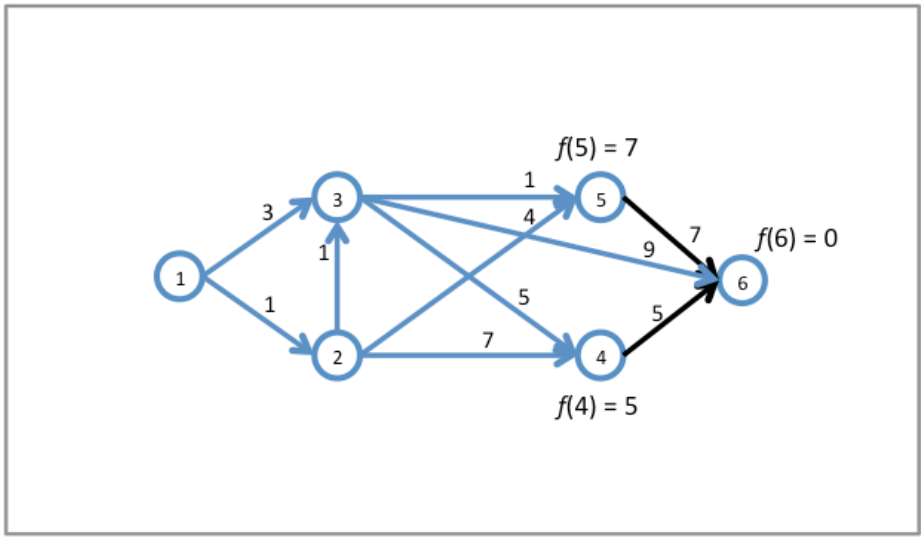

Figure 4-2 After Evaluating Three Nodes

Working our back in reverse order, the next node is 3. Here matters are little busier as node 3 has three nodes in its forward arc: nodes 4, 5, and 6. We outline and evaluate these choices via the following table.

Table 4-1 Evaluating Bellman's Equation at Node 3

j	$d(i, j)$	$f(j)$	$d(i, j) + f(j)$	$f(i)$
4	5	5	10	-
5	1	7	8	**8**
6	9	0	9	-

The shortest path from node 3 is to go through node 5, so that the total distance from node 3 to node 6 is 8. The arc (3, 6) offered a direct path but the length was longer in total than going through node 5. Updating our graph we get the following.

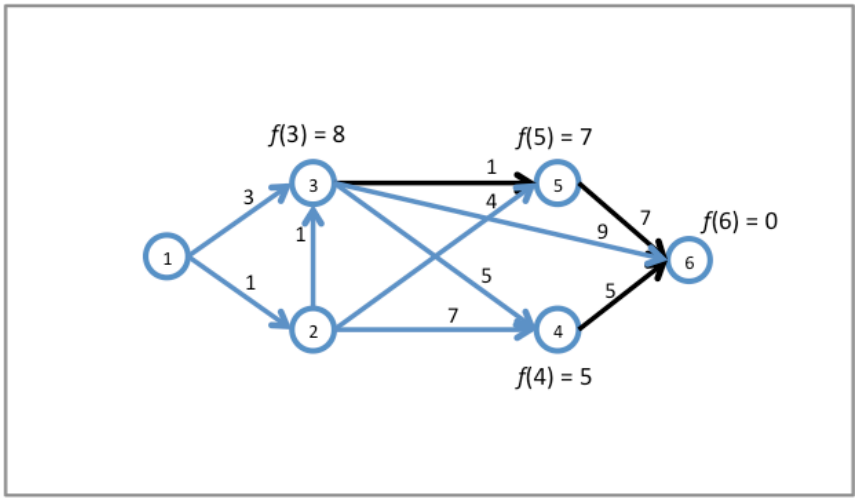

Figure 4-3 After Evaluating Four Nodes

The last table illustrates the algorithm for the remaining nodes.

Table 4-2 Evaluating Bellman's Equation For All Nodes in the Example Network

i	*j*	*d*(*i*, *j*)	*f*(*j*)	*d*(*i*, *j*) + *f*(*j*)	*f*(*i*)
6	na	na	na	na	**0**
5	6	7	0	7	**7**
4	6	5	0	5	**5**
3	4	5	5	10	-
3	5	1	7	8	**8**
3	6	9	0	9	-
2	3	1	8	9	**9**
2	4	7	5	12	-
2	5	4	7	11	-
1	2	1	9	10	**10**
1	3	3	8	11	

The shortest path from node 1 to 6 has length 10. Walking back up the table, we find the shortest path: 1-2-3-5-6. When we update our example network, we see visually the resulting shortest paths tree rooted in the sink.

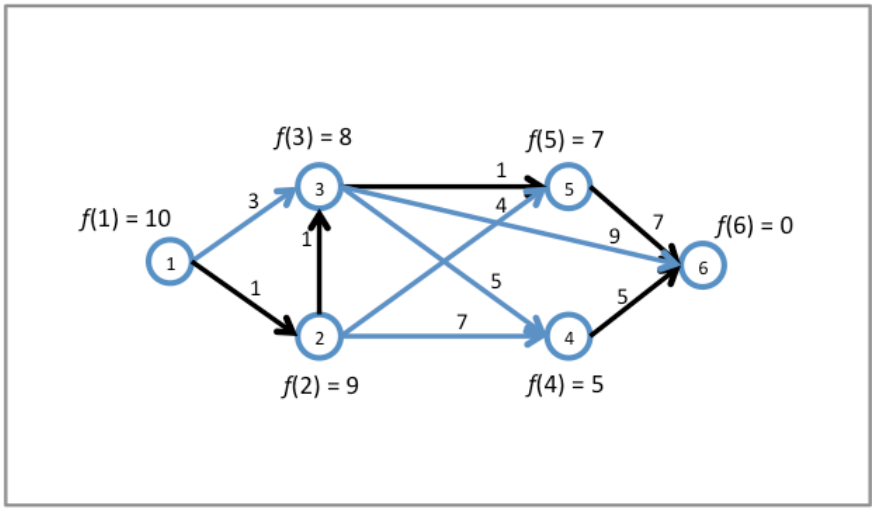

Figure 4-4 Shortest Path Tree

On the term "Functional Equation". Bellman chose the term "functional" from the Calculus of Variations. In the Calculus of Variations a functional is a mapping from a set of functions to a real number. Frequently, the functional is looking for the extreme boundary of a space occupied by a set of functions; i.e., it is functioning as a max or min operator. Returning to our simple example of the shortest path in a network, a function is a mapping from a given point of *i* to the set of path lengths from *i* to the sink. The functional is the mapping from that set of functions to length of the shortest path.

Computational Efficiency As Compared to Enumeration. How much work was saved by employing this procedure? The basic steps involved starting with the sink and going backwards, visiting every node just once, and evaluating its forward arc. In our DP approach, we evaluate each arc once. For this network that is 10 arcs. While with a little bit of stubby pencil work, the reader can determine through brute force

enumeration that there are eight unique paths, the work to find those paths is much more than what DP required. Clearly every arc had to be visited more than once..

What of a general case? Instinct correctly tells us this disparity gets much worse as the problem gets bigger. In a topologically ordered, acyclic direct network, DP visits each arc once. In the worst case of a highly connected network, the source node has n-1 forward arcs, the next node has n-2, and so on, which sum to (n-1)(n-2)/2 arcs, an upper bound of order n^2. Using the same worst case analysis to evaluate the enumeration of paths, at the source we have n-1 choices, at node 2 there are n-2 choices, and so on. Path enumeration requires multiplying these choices, so the number of paths is bounded by (n-1)! which grows much faster than n^2

Longest Path Problem Acyclic Directed Networks. As described earlier, the shortest path problem for acyclic directed networks solves in linear time. What of finding the longest path in the network? For the acyclic directed network, the longest path problem solves in exactly the same fashion as the shortest path problem. We need only take the network and negate all the arc values and solve the shortest path as done before (Ahuja, Magnanti, and Orlin, 1993). Our standard DP approach will work for both minimization and maximization as long as we continue to employ acyclic directed networks. For many sequential decision problems, this requirement is easily met.

Stages and States. In modeling, the nodes represent the possible states of the system, commonly indexed as s and $f(s)$ represents the value of being in state s. The assumption employed in our simple example of states being indexed in topological order,

so that an arc (i,j) would always have $i < j$, is not difficult to satisfy in many sequential decision problems. In fact, a common technique for sequential type problems is to partition the state space into *stages* and *states.* States, indexed by s, can be used to represent the resource level of the system. Stages, indexed by t, can be thought of as time periods through which the system travels while occupying a single state within a time period. This results in the nodes being indexed by the ordered pair (t, s). This imposes a special structure where the system will typically only transition from (t, s) to $(t + 1, s')$ where s may or may not equal s'. This kind of structure is particularly applicable to the binary knapsack problem, which we visit next.

4.2 Dynamic Programming Knapsack Foundations

The binary knapsack provides a classic (albeit simplistic) example of the peacetime capital budgeting problem. As we described previously, the WPSP is different from the peacetime problem in that decisions are made sequentially. The peacetime problem focuses on a long-range horizon so that all options can be considered concurrently. It has its own diverse complications: different resource pools (appropriations), binning of options by type of applications (capability area), inter-relations of projects, and multi-period effects. It is useful nonetheless to consider the simple problem of the binary knapsack since its DP implementation structurally provides a fundamental template on how to approach the more challenging sequential problem.

4.2.1 Binary Knapsack Formulation

The "knapsack" problem gets its name from the challenge any hiker faces while planning his or her trip: there are more items which the hiker would like to take than there is capacity in the knapsack to take them. The hiker needs to decide what unit to attach to the capacity – is the knapsack limited by volume or by weight. Of course, using volume assumes that there is no lost space when packing items in the pack. Many versions of this problem exist in the literature. Prominent in the DP literature (Denardo) is the Integer Knapsack, where the hiker is allowed to choose integer quantities from a set of potential items, subject to the total capacity of the knapsack.

The version we examine here is the binary or 0-1 knapsack. Here each possible item is a "yes/no" option. We ignore interactions to keep things simple: clearly the benefit to the hiker of a space blanket is different whether or not the hiker decides to also bring a sleeping bag.

Relating this to the WPSP, we first examine the case where we know there are T arriving initiatives, we know the order of arrival, and by some oracle all of their associated costs and values are known *a priori.* This is simply the binary knapsack problem with the following structure. The "capacity" of this knapsack is the amount of fiscal resource available.

Deterministic Binary Knapsack Integer Programming Formulation

$$\max \quad \boldsymbol{vx}$$

$$\text{s.t.} \quad \boldsymbol{cx} \leq \mathrm{B}$$

$$\boldsymbol{x} \in \{0,1\}$$

The vector $\boldsymbol{v}$ provides the benefit or value of each item, the vector $\boldsymbol{c}$ provides the cost of each item, and $\boldsymbol{x}$ is the vector of "yes/no" decisions. B is the resource budget available to use. Note that in contrast to the shortest path problem in the previous section, we are now maximizing, but since we will show how we can represent this problem as an acyclic directed network, this use the exact same algorithm.

On the choice of labels: The literature varies, so in this case we have chosen to use the word "value" to indicate the reward or benefit offered by an arriving initiative. We use c to indicate the cost of the arrival, and $f(s)$ as the optimal cumulative value function the system can accrue in state s. We use v and c in this fashion because it is more consistent with terms used by the decision makers we have encountered.

Powell (2007) uses $V(S)$ for $f(S)$ as the value to system of being in state S, and uses v as a sample of V. For reward or benefit he uses C for contribution. Denardo (2003) uses v for cumulative reward or value, $f()$ for the optimal v, and uses R for the reward. We define the dynamic program for the binary knapsack as follows.

Deterministic Binary Knapsack Recursive Dynamic Programming Formulation

Stages

t the t^{th} initiative arriving, $t = \{1,\ldots,T\}$

States

b_t remaining budget when the t^{th} initiative arrives (perhaps in units of \$M)

Data

c_t cost of initiative t

v_t value of initiative t

B total budget at the start of the fiscal period

Variable

x_t 1 if the initiative t is chosen for the portfolio, 0 otherwise

Value Function

$f_t(b_t)$ – maximum cumulative expected value earned from initiatives $t, t+1,\ldots, T$

Functional Equation

$$f_t(b_t) = \max_x \{v_t x_t + f_{t+1}(b_t - c_t x_t)\}$$

Constraints (also called Boundary Conditions)

$$b_t \geq c_t x_t \geq 0$$

$$b_1 = \mathrm{B}$$

$$f_{T+1}(b) = 0, \text{ for all } b$$

The recursion asks "Is it better to step forward with the arrival, and decrement the budget accordingly, or to step forward without the arrival and keep more of the budget?"

A note on the choice of t as the stage/initiative index: Many formulations will use the common i but given the sequential nature of arrivals – whether deterministic or random – the use of t links the stages and arrivals to the notion of time advancing.

As consistent with our simpler earlier example, this formulation defines an acyclic directed network. Each node (t, b_t) maps to a stage t for the current arrival and a state b_t for the amount of remaining budget. We define a source node for a stage 0 and state B that represents the system prior to any arrivals, and a sink node w after all stages in order represent the system after all decisions are done. From the source node we create single arc of length 0 signifying the first arrival. For all stages $0 < t <= T$, each node (t, b_t) has two forward arcs only, one to node (t+1, b_t-c_t), depicting a "yes" decision with the budget decremented by c_t, and (t+1, b_t), a "no" decision with an unchanged budget. The former forward arc has value v_t and the latter has zero value. All nodes (T+1, b_{t+1}), for $0 <= b <= B$, represent a dummy stage to capture the results of the last initiative considered. From each dummy stage node (T+1, b_{t+1}) we create a single forward arc of zero length to the sink. Solving the dynamic program equates to finding the shortest path from every node to sink in this network. Another example will help illustrate.

4.2.2 Binary Knapsack Example

In this example, there are five stages, representing four separate arriving initiatives and final dummy state, and six units of resource. The table below provides the data for the four initiatives.

Table 4-3 Data for Example Knapsack

Initiative t	Cost c_t	Value v_t
1	4	6
2	3	4
3	2	3
4	1	3

The cost data is in the same units as our resource budget. The value data is in its own units. Using these data we construct the following network.

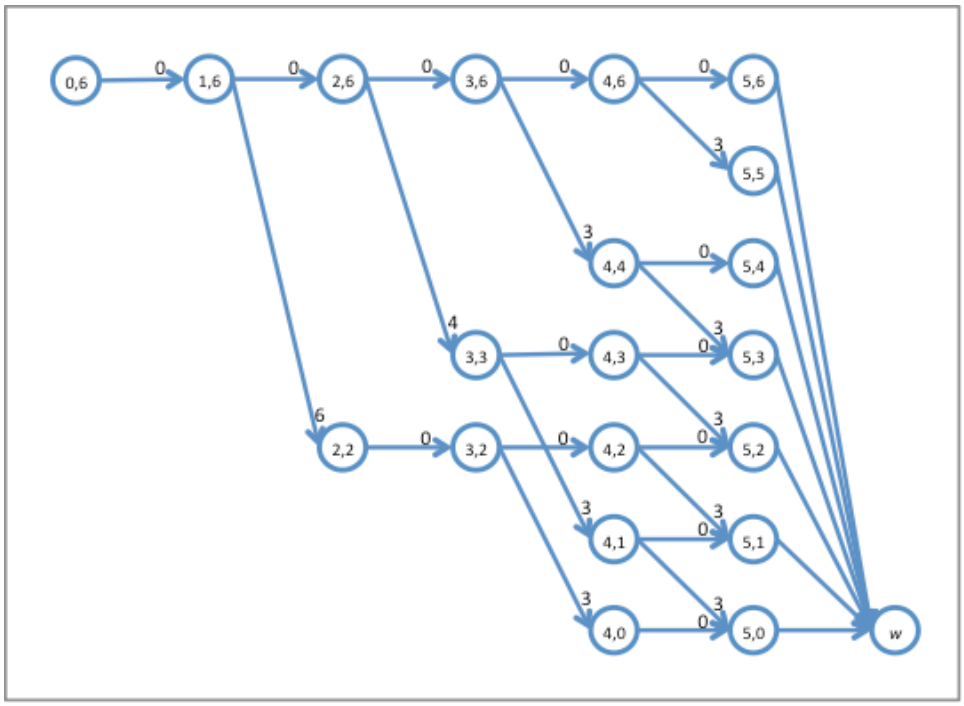

Figure 4-5 Example Knapsack Network

We note that the network is relatively sparse. Not all states *b* can be occupied in all stages, so for clarity we have removed these infeasible nodes. Transitions only occur from state *t* to state *t+1*. The forward arc of each node – except after the last stage when there are no remaining decisions – is binary, representing the "yes/no" decision. The "down" arcs represent a "yes" decision, indicating a reduction in the budget state. Because each stage *t* corresponds to each initiative, and each initiative has its own cost, the "slope" of the down arcs represents the change in budget. Not every budget state is feasible at every stage nor is every decision available at every feasible state/stage. The arc length data is v_t, the value gained by traversing the arc. The cost c_t of an initiative appears in the change in budget state, so that when we exchange resources for the value

offered by initiative t the system moves from state b_t to state b_t-c_t and from stage t to t+1. This network is clearly acyclic and its readily evident that we could easily re-label the nodes so that $i < j$ for all (i,j).

We take a brief tour of the network. From the source node of (0,6) we step directly to (1,6) where our first consideration is whether to acquire initiative 1. We can choose not to do so, and arrive straight at node (2,6). If we choose to acquire initiative 1, we pay a cost of 4 units, gain 6 units of value, and arrive at node (2,2). Consider now initiative 2. It can only be acquired if we decided not to acquire initiative 1, since there's not room enough in the budget for the two of them. Initiative 3 can be acquired if we had not acquired either 1 or 2, or if we acquired either. Finally, we can acquire initiative 4 under all circumstances except where we acquire 1 and 3, since acquiring these two initiatives leaves no resources for initiative 4.

Before we apply DP to the network, we should address a possible condition not previously discussed: multiple optima, or how to break ties. Multiple optima in this case occur when either binary decision offers the same value; that is, $v_t + f_{t+1}(b_t - c_t) = f_{t+1}(b_t)$. In general, a rule is required to determine how to break ties. By the recursion, it does not matter which choice is made. The chosen rule depends upon the context and the bias of the decision makers. A capability developer may have a bias for making acquisitions – "a bird in the hand is worth two in the bush", while the financial manager may want to "save for a rainy day". For our example we will follow the latter policy for breaking ties.

The following illustrations show the algorithm after solving for all nodes in each stage moving backwards from the sink.

The first stage solved is the dummy state of T+1, or stage 5. This stage is actually a boundary condition and involves setting each dummy stage distance from the sink to 0. Now this boundary condition might seem peculiar, since intuition for many might be that we should arrive at the sink having gained the value of the decisions represented by the path that brought us there. DP employs recursive thinking, where it assigns the value to the node based on its distance from the sink – the inherent value the system gains by being in that state. An analogy is to consider the concept of potential energy for those familiar with Physics.

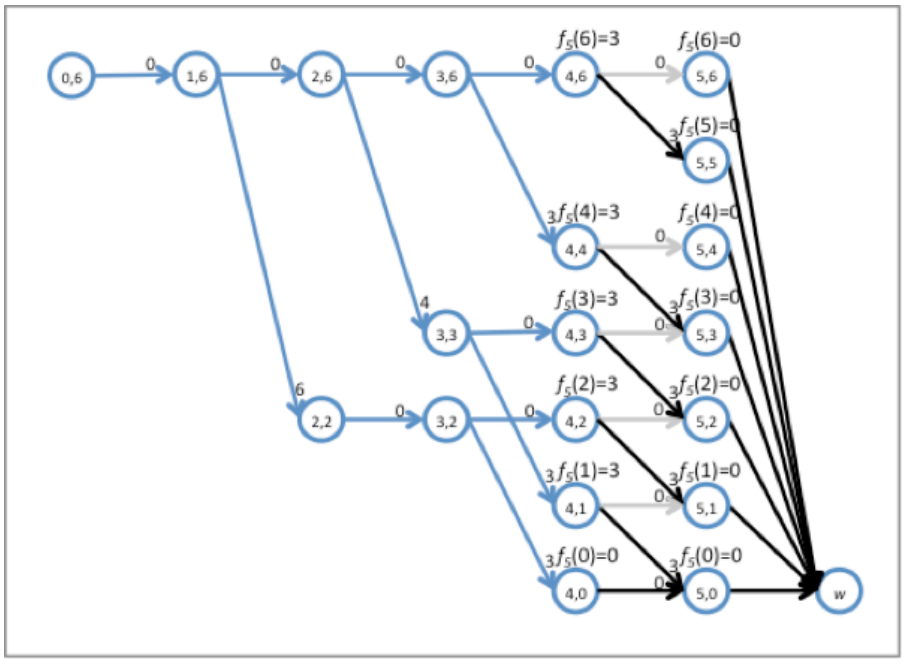

Figure 4-6 Example Knapsack Network After Solving for Stage 4

The dark arcs represent the chosen paths for this stage and the gray are the sub-optimal arcs. Note that for this stage at all states the choice was to acquire initiative 4 except at (4,0) because there were no remaining funds.

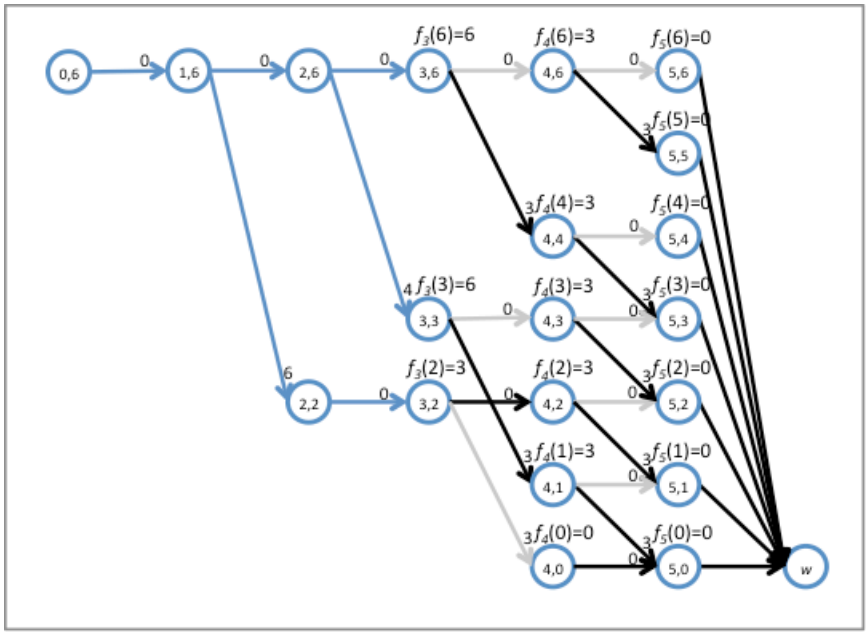

Figure 4-7 Example Knapsack Network After Solving for Stage 3

During stage 3 with the system at resource state 2, we faced a tie: we could acquire initiative 3, gain 3 units of value plus whatever value was at node (4,0); or we could forego initiative 3, gain no value but still have whatever value was at node (4,2). Node (4,0) has zero value while node (4,2) already has 3 units of value. These two

options were tied in terms of the resulting distance from the sink, however, our tie breaker dictated that we choose the path that gives us the same value for less (or 0) resources. Thus the arc ((3,2),(4,2)) is darkened indicating the chosen path.

Continuing this logic backwards to remains nodes, we see the resulting shortest path tree from every node to the sink.

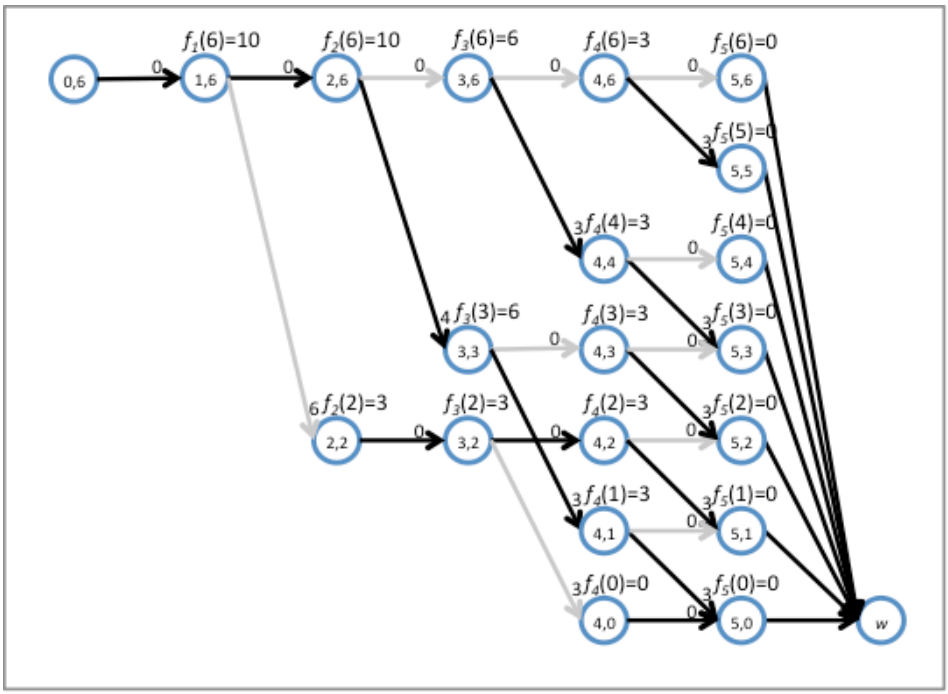

Figure 4-8 Example Knapsack NetworkAfter Solving for All Stages

As mentioned previously the order of the initiatives does not matter. If we change the order of the initiatives in generating the network, the result may look different visually but the results will be identical since we are solving over the same fundamental set of paths. Solving the knapsack problem requires generating, explicitly or implicitly, the set of all feasible combinations of items and finding the subset of these with the maximum summed value. Since summation is commutative, the order of the items in the subset is of no consequence.

4.2.3 Some Helpful Concepts

Bellman's Principal of Optimality. As stated earlier, DP breaks the larger, seemingly intractable problem into smaller sub-problems. Bellman states (1957, Chap. III, pg. 3):

> Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Denardo (2003, Chap. 2, pg 15) simplifies this Principle of Optimality for the case of the shortest path in a network:

> Consider an optimal path from some node to some other node. Any path $(i_p,\ldots,i_q)$ contained in this path is an optimal path from node i_p to node i_q.

Recursive DP builds this optimal path one arc at a time, stepping backward one node at a time and then looking forward from each node it visits. The binary knapsack emphasizes the simplicity of the choices that are at the heart of recursive DP: make decisions based upon the best marginal improvement in state. Trading resources for value at stage t will lead to a lower *potential* for future value. Not trading those resources keeps a higher *potential* – but does not acquire any further value, which runs counter to goal of the program. Thus the decision must be made on the *marginal improvement* in value: so that only if $v_t + f_{t+1}(b_t - c_t) - f_{t+1}(b_t) > 0$ will we acquire the t-th item. This is essentially the *reduced cost* of the t-th item given budget level b, where $x_t = 1$ iff the reduced cost of the item t is greater than 0. We can also consider what Papastavrou, Rajagopalan, and Kleywegt (1996) call a "critical reward" level at resource state b at stage t for an item of cost c.

Equation 4-2: Critical Reward Equation

$$R_t^b(c) = f_{t+1}(b_t) - f_{t+1}(b_t - c_t), \text{ if } b_t \geq c_t, \text{ else } R_t^b(c) = \infty.$$

$$\text{If } v_t > R_t^b(c) \text{ then } x_t = 1, \text{ and } x_t = 0 \text{ otherwise.}$$

Policy. Looking at the DP formulation it is easy to think about the optimal decision as existing in isolation. The concept of a policy provides the general notion that

one can consider decisions as belonging to a rule set. Echoing Denardo (2003), in DP, a policy is a function that maps to each state (or, the combination of state and stage) a decision. In the context of the shortest path problem, it dictates for every node *i* the choice of node *j* in its forward arc. In the context of the binary knapsack, a policy would tell us what "yes/no" decision to make for a given budget level *b* and stage *t.* Mathematically, we can describe a policy for the binary knapsack as follows.

Equation 4-3: Binary Knapsack Policy

$$\boldsymbol{\pi} = \{x_{b,t}:\ (b, t) \rightarrow x_{b,t}, x_{b.t} \in \{0,1\}\}$$

When considering our shortest route problem over an acyclic network, a policy would describe a tree. The optimal policy $\boldsymbol{\pi}^*$ is the policy that dictates the optimal choice of forward arc for every node *i*; i.e., the shortest path tree. Lastly, while we have described policies in terms of the variable x_t, when considering the binary nature of the decision we can think also define our policy in terms of the choice of arc based on a rule such as "Acquire item *t* if $v_t > R_t^b(c)$; reject otherwise."

Post-decision State. Each node represents the consideration the decision maker gives to initiative *t* in light of the remaining budget b_t. The decision maker's decision then carries the system to the next stage/state. Though not necessary for the static binary

knapsack, we could easily partition what is happening in each stage into two kinds of states still indexed by resource level: the pre-decision state where the decision maker is considering his or her choice, and the post-decision state which shows the instantaneous effect of the decision when the system has not yet advanced to next stage. For this static binary knapsack, the post-decision state would have a single arc straight to the next stage's corresponding resource state. The transition is deterministic. The post-decision state becomes very important later when begin to implement machine learning.

Horizon Assumptions and Discounting. In addressing many decision problems with sequential decisions one must consider the decision epoch and its endpoint: the decision horizon. Is there a point in the future at which we are "done"? Or does the problem require one to consider the implications of decisions to a point arbitrarily far into the future? This question leads to a natural partition of approaches between those with finite and infinite horizons. Since federal agencies are required to commit all of their funds within a fixed period, most commonly in the same year they are issued, we will focus on the finite horizon assumption. We will use the term "decision epoch" to represent the time period from the start to the "horizon", beyond which no decisions can be made nor value accrued.

4.3 Stochastic Binary Knapsacks

A point of emphasis for Powell (2007) in presenting stochastic DP is where in the decision process the previously random information becomes known to the decision maker. He illustrates this point in discussing a traveler through a road network. Powell

describes a driver who must choose at an intersection which road to turn on, but who will not know the exact travel time until having traversed the road. The driver has a long history of travel on the possible paths and knows the distribution of travel times across each road. Thus, the driver uses the prior history to make the decision. The functional equation of the decision making process may look as follows:

Equation 4-4: Stochastic Functional Equation – Type I Stochastic Knapsack

$$f(i) = \max_j \mathbf{E}\{c_{ij} + f(j)\}$$

We will call this the *Type I Stochastic Knapsack*. The equation captures the notion that we are making decisions at each node based upon on an expectation of the travel time. The other case Powell describes is one where at each intersection the driver is told with certainty the travel time for the choices. Perhaps, they can look at a GPS navigation system which gives the driver a very accurate "real time" prediction of travel time across each choice. The latter problem, which we call the *Type II Stochastic Knapsack*, has a subtly different form.

Equation 4-5: Stochastic Functional Equation – Type II Stochastic Knapsack

$$f(i) = \mathbf{E}\{\max_j (c_{ij} + f(j))\}$$

The expectation operator is now outside the maximization operator because *a priori* the decision itself is random. Thus, while in both cases the distribution is assumed known, in Type I problems we will arrive at an invariant set of decisions: each time we travel the network we make a decision based on the same expectation about our choices. In Type II problems, at each node we are presented with a single sample from the probability distribution. In the latter case, a policy would have to consider many more possibilities. The Type II problem is our focus since it more closely mimics the situation in the WPSP. However, it is instructive to start with the Type I static knapsack.

4.3.1 Type I Stochastic Binary Knapsack Variations

Random Initiative Values[3]. At a given state, the value of the initiative has a random distribution but the result of the binary decision is known deterministically, since the state is the current budget level, and the cost of choosing an initiative is known. The choice is to reject and stay put at the current resource state level and then advance to the next stage, or accept and move to a known lower resource state for a random value and advance to the next stage. The goal is to maximize the expectation of the total reward over the random vector of initiative values $\boldsymbol{v}$.

[3] We use "initiative value" with care since the term *value function* has a special meaning in dynamic programming.

Static Stochastic Binary Knapsack with Random Benefits Integer Programming Formulation

$$\begin{aligned} \max \quad & E[\boldsymbol{vx}] \\ \text{s.t.} \quad & \boldsymbol{cx} \leq B \\ & \boldsymbol{x} \in \{0,1\} \end{aligned}$$

This is not to be confused with an alternative interpretation of the static stochastic knapsack, where the goal is to maximize the probability that the value of the solution exceeds a deterministic threshold *e*.

Static Stochastic Binary Knapsack with Random Benefits and Chance Objective Integer Programming Formulation

$$\begin{aligned} \max \quad & \Pr(\boldsymbol{vx} > e) \\ \text{s.t.} \quad & \boldsymbol{cx} \leq B \\ & \boldsymbol{x} \in \{0,1\} \end{aligned}$$

For an examination of this version of the stochastic knapsack problem and some solution approaches, see Morton and Wood (1998).

We introduce the following recursion formulation. Since the results of our decisions depend on samples from probability distributions, the value function $f_t(b_t)$ is no longer deterministic but is now an expectation.

Static Stochastic Binary Knapsack with Random Benefits Recursive Dynamic Programming Formulation

Stages

t the t^{th} initiative arriving, $t = \{1,\ldots,T\}$

States

b_t remaining budget when the t^{th} initiative arrives (perhaps in units of \$M)

Deterministic Data

c_t cost of initiative t

B total budget at the start of the fiscal period

Random data

$\boldsymbol{v}_t$ stochastic benefit of initiative t

Variable

x_t 1 if the initiative t is chosen for the portfolio, 0 otherwise

Value Function

$f_t(b_t)$ – maximum expected reward that can be earned from initiatives t, t+1,…, T

Functional Equation

$$f_t(b_t) = \max_x \mathbf{E}[\{\boldsymbol{v}_t x_t + f_{t+1}(b_t - c_t x_t)\}]$$

Constraints (also called Boundary Conditions)

$$b_t \geq c_t x_t \geq 0$$

$$b_1 = \mathrm{B}$$

$$f_{T+1}(b) = 0, \text{ for all } b$$

Once the decision x is made, the next state is known deterministically because only the values are stochastic, while the cost of the arrival, which affects the resulting state, is deterministic. The value or benefit of initiative t does not depend upon the action x and the expectation of an expectation is the same, so we can rewrite this recursion.

Equation 4-6: Type I Stochastic Knapsack with Random Benefits

$$f_t(b_t) = \max_x \{\mathbf{E}[V_t]x_t + f_{t+1}(b_t - c_t x_t)\}$$

We can rewrite the expectation in the case of a discrete probability distribution as follows, where j represents the possible values that $\boldsymbol{v}_t$ can assume.

Equation 4-7: Type I Stochastic Knapsack with Discrete Random Benefits

$$f_t(b_t) = \max_x \{ [\sum_j \Pr(V_t = v_t(j))\, v_t(j)]\, x_t + f_{t+1}(b_t - c_t x_t) \}$$

This Type I Stochastic Knapsack with Random Benefits is ultimately not fundamentally different than our previous case. We have only replaced the value of the initiative, which was previously given, with its expectation. Note that in the case of the discrete random benefits as shown in the equation above, the summation plays no role in the decision. The figure below illustrates the choices.

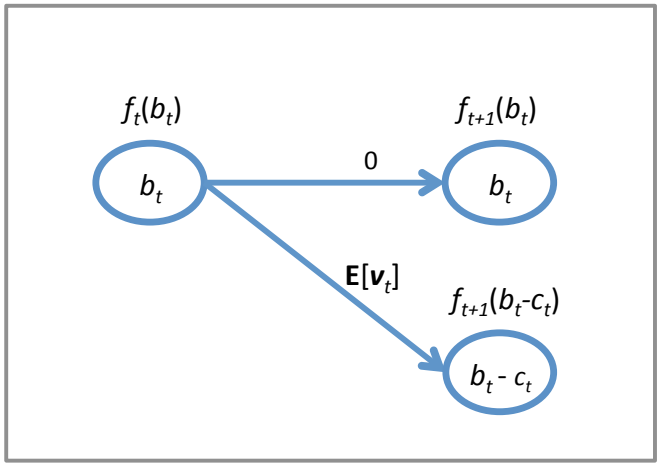

Figure 4-9 Graphical Depiction of Bellman's Equation for Type I Stochastic Knapsack with Random Benefits and Fixed Costs

The binary nature of the decision allows us to express this decision problem very simply from a policy perspective.

Equation 4-8: Type I Stochastic Knapsack with Random Benefits Policy

$$\pi^*(b,t) = \begin{cases} \text{accept if } \mathbf{E}[V_t] + f_{t+1}(b_t - c_t) > f_{t+1}(b_t), \text{ and } c_t <= b_t; \\ \text{reject otherwise.} \end{cases}$$

Note that the policy formulation makes explicit what choice we make in the event of a tie: we will accept only if the change in value is strictly improving. Had the inequality been greater than or equal, then we would accept the new initiative in the event of a tie.

This is the sample problem described by Powell (2007) of a driver traveling the road network who at each intersection must choose which turn to take without knowing precisely the travel time across each of the road segments. In this formulation, drivers would make their decisions based on the *expected travel time* of each segment.

Binary Knapsack Variation with Random Costs. Now consider the alternative where at each node the values are known but the costs are stochastic. The goal is to maximize the expectation of the total reward over the random vector of initiative costs $\boldsymbol{c}$. This is also a Type I problem so we are making our decisions knowing only the

distribution of cost; only sometime after making our choice will we learn the *true* cost of the item. At first blush this case is more challenging, because while the benefit is known, the next state is random since the resource level defines the state of the system. In particular, it creates a special problem in that the decision maker cannot incur a cost greater than the available resource budget. We present a modification of the previous formulation, showing only the changes. All other definitions remain the same.

Static Stochastic Binary Knapsack with Random Costs Recursive Dynamic Programming Truncated Formulation

Random Variables

$\boldsymbol{c_t}$ random cost of initiative t

Recursion

$f_t(b_t) = \max_x \mathrm{E}[\{v_t x_t + f_{t+1}(b_t - C_t x_t)\}]$

Constraint

$\mathbf{Pr}(C_t > b_t) \leq \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is a small non-negative number.

One might ask: When would a decision maker decide to make an expenditure without knowing the resulting costs? The answer is that there are many situations where costs are only estimated and change over time. In any research and development

situation, the costs, eventual value and timing are all estimates. The decision maker has to make decisions about what choices to make at each stage of development based on only partial information and estimates of expectations: essentially the decision maker must decide what research and development path to follow - in order to maximize the yield of his programs while staying within budget. In some cases, managers may be able to go back to leadership to request more funding if needed, but this can expose their program to cancellation. One can think of **ε** as representing the level of risk the decision maker is willing to take in overspending the allocated budget.

In this particular problem, the choices are binary: choose not to acquire and suffer no change to your resource level, or acquire the fixed value but the resources consumed are uncertain. Assuming the cost of initiative *i* is (or can be approximated as) discretely distributed with a total *J* of possible values $c_t(j)$ and probabilities $\Pr(C_t = c_t(j))$, we can express this recursion as follows.

Equation 4-9: Type I Stochastic Knapsack with Random Costs

$$f_t(b_t) = \max_x \{v_t x_t + [\Sigma_j \Pr(C_t = c_t(j)) f_{t+1}(b_t - c_t(j)x_t)]\}$$

While in the random benefits case we were able to remove the decision variable from within the expectation operator, this is not the case here.

We can also express this from a policy perspective.

Equation 4-10: Type I Stochastic Knapsack with Random Costs Policy

$$\boldsymbol{\pi}^*(b,t) =$$

$$\begin{cases} \text{accept if } v_t + [\sum_j \Pr(C_t = c_t(j))\, f_{t+1}(b_t - c_t(j)] > f_{t+1}(b_t), \text{ and } \Pr(C_t > b_t) \leq \boldsymbol{\varepsilon}; \\ \text{reject otherwise.} \end{cases}$$

The policy perspective emphasizes both the overall expected cost of the decision and its functional nature. Even so, for every state there is a simple rule that guides the decision maker.

Graphically this requires a change in the structure of the network: we now need to partition the layers into decision nodes and chance nodes, as depicted below. The decision nodes lead to chances nodes. The chances nodes are labeled by the actions, indexed by state and stage, that brought the decision maker to the current state. The chance nodes carry the expected value, which is the inner product of the probabilities and the value of the states in the forward arc. When calculating value, the arcs from the decision node have additive value. The arcs from the chance nodes are weighted with their corresponding likelihood, so that they function in multiplicative fashion. Powell (2007) also describes these chance nodes as the Post-Decision States.

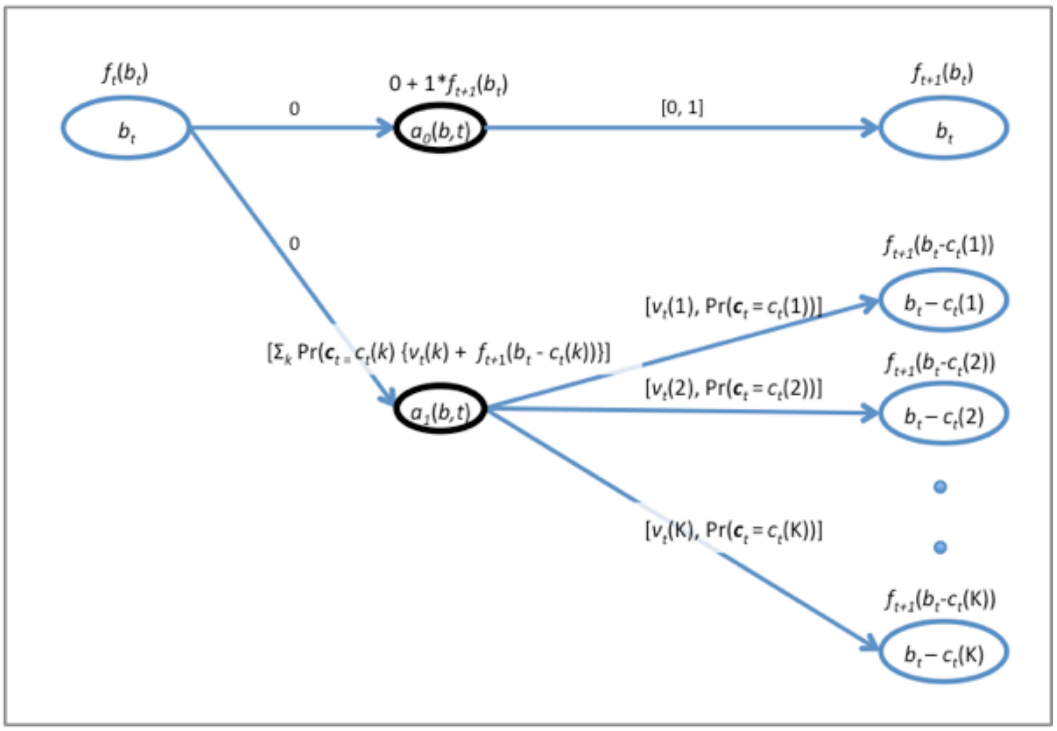

Figure 4-10 Post-decision State for the Type I Stochastic Knapsack with Random Costs

This case is actually an example of a Markov Decision Process (MDP): the decision made then leads to a respective set of transition probabilities. We will give MDPs only cursory treatment here. While they are important to what we classify as Type I stochastic knapsacks, our focus is on the Type II problem. In the Type II problem, the decision is random while the transition is deterministic.

Static Stochastic Binary Knapsack Variation: Random Benefits and Costs. We now consider the case where at each node both the benefit and the cost are stochastic. Furthermore assume these are independent of each other. The independence assumption

allows us to move the expectation outside of the summation, as was the case for the stochastic binary knapsack with random benefits.

Equation 4-11: Type I Stochastic Knapsack with Independent Random Benefits and Costs

$f_t(b_t) = \max_x \mathbf{E}[\{V_t x_t + f_{i+1}(b_i - C_t x_t x_i)\}]$, which can be rewritten:

$f_t(b_t) = \max_x \{\mathbf{E}[V_t]x_t + \Sigma_j \Pr(C_t = c_t(j)) f_{t+1}(b_t - c_t(j)x_t)\})\}$, where $\Pr(C_t > b_t) \leq \boldsymbol{\varepsilon}$.

Mathematically, this gives us the same basic situation as shown when reward was fixed and cost random, where all that we have done is replaced the fixed cost with its expected value.

Suppose that cost and benefit come from a joint distribution where independence cannot be assumed. In fact, independence would be an *unnatural* assumption in any kind of market where one would expect that benefit and cost are correlated. If we can discretize the joint distribution for K values, we get the following functional equation.

Equation 4-12: Type I Stochastic Knapsack with Dependent Random Benefits and Costs

$f_t(b_t) = \max_x \Sigma_k \Pr(C_t = c_t(k), V_t = v_t(k)) \{v_t(k)x_t + f_{t+1}(b_t - c_t(k)x_t)\})\}$

Depicting this from a policy perspective clarifies some of the choices involved.

Equation 4-13: Type I Stochastic Knapsack with Dependent Random Benefits and Costs Policy

$$\pi^*(b,t) =$$

$$\begin{cases} \text{accept if } \Sigma_k \Pr(C_t = c_t(k),\ V_t = v_t(k))\ \{v_t(k) + f_{t+1}(b_t - c_t(k))\} > f_{t+1}(b_t), \\ \qquad\qquad \text{and } \Pr(C_t > b_t) \le \varepsilon; \\ \text{reject otherwise.} \end{cases}$$

This is not qualitatively different than our previous policy statement. Depicting this graphically we see again that we have had to change the structure of the network, this time in how we handle the data on the arcs.

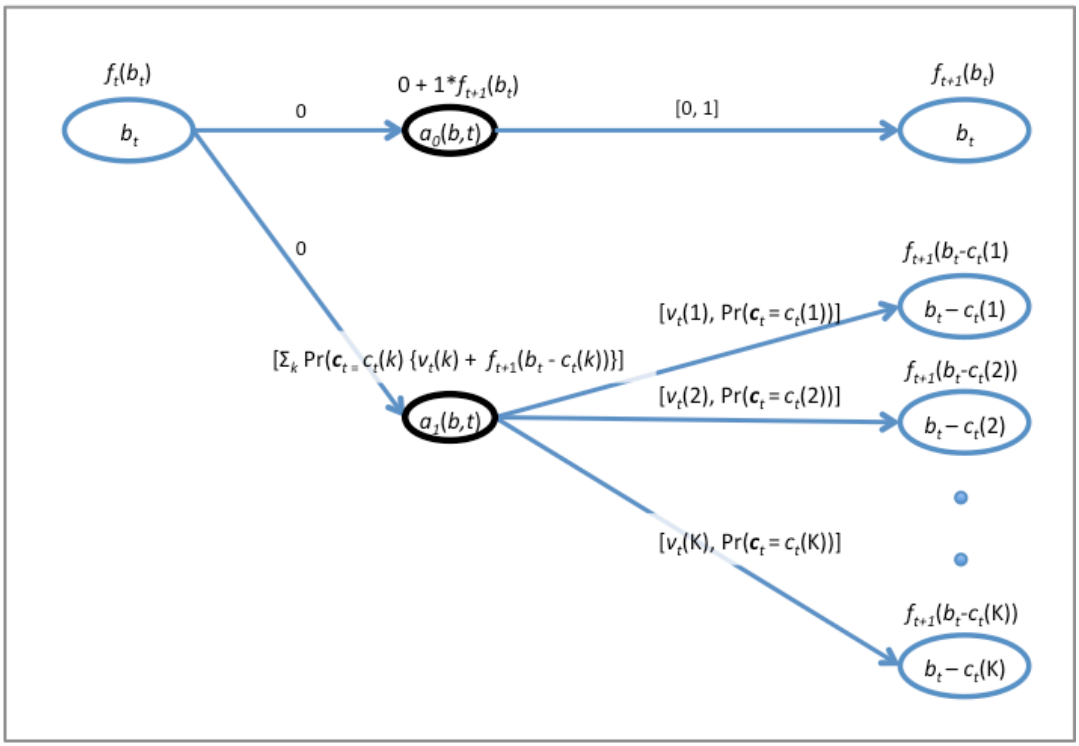

Figure 4-11 Post-decision State for the Type I Stochastic Knapsack with Random Benefits and Costs

There is no immediate reward after the decision, thus each of these arcs have reward 0. The forward arcs from the post-decision nodes contain an ordered pair: the additive reward and multiplicative weight assigned to this outcome.

The chance nodes contain the value of the decision, which is now the weighted sum of jointly dependent reward and the value of the future state; i.e., the expected value of the decision.

In the Type I problem, for a given problem the optimal policy is invariant with respect to the resource level and stage (b,t). At each node the decision is made based

upon the expectation of the random variables. Assuming perfect knowledge of the distribution of travel times and that these are distributions are stationary, perfectly rational drivers traversing the network regularly will make the same decision at every node even if some days the travel time is slower than expected because they know that in the long run they are following an optimal policy.

We will now turn our attention to the Type II problem, where the analogy is that our perfectly rational driver traversing the same road network under the same conditions has now been given a perfect GPS-based oracle that tells the driver how much time it will take to cross each roads segment emanating from a given intersection. Perhaps the choice with the best expected value today has an accident a half-mile down the road. Thus, Type II problems will result in dynamic policies with respect to (b,t), by which we mean that the policy may vary with each trip through the network. As we will see, the policies are not dynamic but rather that they require a larger concept of system state, not just (b,t) but also incorporating the new information presented to the decision maker when arriving at the node.

4.3.2 Dynamic Stochastic Binary Knapsack

We introduce a new feature that distinguishes the Wartime Defense Portfolio Problem from previous binary stochastic knapsacks: the random number of arriving initiatives. This creates a problem for creating the network depiction. However, we will see how to adapt the binary knapsack DP approach to handle this challenge.

The approach described here is due to Papastavrou, Rajapopalan, and Kleywegt (1996). Items arrive over a finite horizon of discrete time periods. Each time period is a Bernoulli trial where a single item shows up with probability p, and no item with probability 1-p. Using Bernoulli trials can be viewed as an approximation to a Poisson arrival process, where time has been discretized into small enough units that the probability of a second arrival can be ignored. This gives us an approach for addressing the unknown number of arrivals.

If a new item arrives, its value and cost instantly become known and an immediate decision must be made whether to acquire or discard. As alluded to before, the decisions *a priori* are random, since we must first establish *if* an initiative has arrived, and, if so, its reward and cost, and the remaining resource budget. Papastavrou, Rajapopalan, and Kleywegt solve such problems via a slightly modified approach to Bellman's, where they must consider four Events that address all possible states at time t:

> Event 1: an initiative has arrived, the arriving initiative has cost less than remaining budget and it meets the optimality conditions.
>
> Event 2: an initiative has arrived, the arriving initiative has cost less than remaining budget, but it does not meet optimality conditions.
>
> Event 3: an initiative has arrived but its cost exceeds the remaining budget.
>
> Event 4: an initiative did not arrive.
>
> Let us first consider this as the authors did from a policy perspective.

Equation 4-14: Type II Stochastic Knapsack Policy Equation

$$\boldsymbol{\pi}^*(b,t) = \begin{cases} \text{accept if } v_t + f_{t+1}(b_t - c_t) > f_{t+1}(b_t), \text{ and } c_t \le b_t; \\ \text{reject otherwise.} \end{cases}$$

This looks identical to the deterministic problem, which we should expect since the nature of the Type II problem is that at the point of the decision, the data on the item has become known; i.e., the decision is conditioned on these data. Using the four events that define the optimal decisions, one can construct the following recursion:

Equation 4-15: Type II Stochastic Knapsack Recursion

$$f_t(b_t) = p\,[$$

Event 1
$$[\Pr(C_t \le b_t,\ V_t + f_{t+1}(b_t - C_t) > f_{t+1}(b_t)]$$
$$\times\ \mathbf{E}[V_t + f_{t+1}(b_t - C_t) |\ C_t \le b_t,\ V_t + f_{t+1}(b_t - C_t) > f_{t+1}(b_t)]$$

Event 2
$$+\ [\Pr(C_t \le b_t,\ V_t + f_{t+1}(b_t - C_t) \le f_{t+1}(b_t)]\, f_{t+1}(b_t)$$

Event 3
$$+\ [\Pr(C_t > b_t]\, f_{t+1}(b_t)]$$

Event 4
$$+\ (1\text{-}p)\, f_{t+1}(b_t)$$

With Boundary Conditions

$$f_{T+1}(b) = 0, \text{ for all } b$$

Consider the events in reverse order. Event 4 says with probability (1-p) no item arrives and the value $f_{t+1}(b_t)$ remains as it was. In Event 3 with probability p an item arrives but with probability $\Pr(C_t > b_t]$ its cost exceeds the available budget so that value $f_{t+1}(b_t)$ remains as it was. In Event 2 an item arrives with probability p and with joint probability $[\Pr(C_t \le b_t,\ V_t + f_{t+1}(b_t - C_t) \le f_{t+1}(b_t)]$ the cost is within budget but the reward is insufficient so value $f_{t+1}(b_t)$ remains as it was.

The sole event in which an acquisition is made is Event 1. Again with probability p an item arrives and with joint probability $[\Pr(C_t \le b_t,\ V_t + f_{t+1}(b_t - C_t) \le f_{t+1}(b_t)]$ the item is acquired. However, the amount of value gained is uncertain since the next state depends upon the cost of the arrival. This value is shown as a conditional expectation, which is simply the average of the values gained over each of the acceptable choices in the forward arc.

Papastavrou, Rajapopalan, and Kleywegt (1996) show how p acts as a filter on the value without changing the behavior of the system qualitatively. Thus, they let $p = 1$ and omit Event 4 from their recursion. We are interested in developing decision support tools so we have retained p in the recursion. The authors describe various conditions in similar fashion to the treatment we gave to the Type I stochastic binary knapsack: fixed cost and variable reward, variable cost and fixed reward, and variable cost and reward.

In the case of variable costs, they discuss how as the deadline approaches, the combinatoric nature of the knapsack problem can result in counter-intuitive results. First

they discuss how one should expect consistent behavior: if an item of fixed value v and cost c is unacceptable at state (b,t) then decreasing the available budget should not result in the item becoming acceptable. They then provide a numerical example where *inconsistent* behavior takes place: the critical cost goes up even as the available budget goes down. They establish certain conditions on the distribution of costs to ensure consistent behavior, in particular that the distribution of cost should be concave and monotonically decreasing on $(0, \infty)$. We will discuss the implications of these conditions later in the context of a numerical example.

When determining how to convert this recursion to an algorithm, or indeed to consider how to efficiently handle a policy space that has leaped from being on (b, t) to being on (b, t, c, v), we turn to the concept of the critical reward, $R_t^b(c)$. The critical reward, a function of *b, t,* and *c,* partitions the reward space between acceptable and unacceptable values. It is defined as below.

Equation 4-16: Type II Stochastic Knapsack – Critical Reward

$$R_t^b(c) = f_{t+1}(b_t) - f_{t+1}(b_t - c_t), \text{ if } b_t \geq c_t, \text{ else } R_t^b(c) = \infty$$

Then for any arrival at time *t,* budget *b,* with cost *c* and value *v*, we define the optimal policy $\boldsymbol{\pi}^*(b, t, c, v)$ as being accept if $v_t > R_t^b(c)$ and reject otherwise. We present how the recursion translates into an algorithm.

DSKP Recursive DP Algorithm

Step 1. Let:

$f_t(b) = 0$ for all b in $[0, B]$ and t in $[1, T+1]$

$\Pr(c, v)$ be probability of arrival with cost c and reward v; $\Sigma_k \Pr(c_t(k), v_t(k)) = 1$

p be probability of an arrival in any given t

Step 2. For t in $[T, 1]$ {

For b in $[B, 0]$ {

For k in $[1, K]$ {

If $c(k) <= b$ then {

$R_t^b(c(k)) = f_{t+1}(b) - f_{t+1}(b - c(k))$

If $v(k) > R_t^b(c(k))$ then

$f_t(b) = f_t(b) + p \Pr(c(k), v(k)) f_{t+1}(b - c(k))$

} else {

$f_t(b) = f_t(b) + p \Pr(c(k), v(k)) f_{t+1}(b)$

}

} else

$f_t(b) = f_t(b) + p \Pr(c(k), v(k)) f_{t+1}(b)$

}

}

$f_t(b) = f_t(b) + (1 - p) f_{t+1}(b)$

}

}

Step 3. Return $f()$ and $R_t^b()$.

Assuming that we have accurately modeled the decision problem, all that is needed to make a decision is $R_t^b(c)$, which can be stored as a table of critical reward values.

WPSP Small Scale Example.

This example seeks to capture the fundamental aspects of JIEDDO's challenges. From Chapter 1, analysis of JIEDDO's initiatives led to two key observations: arrivals could be approximated by a Poisson arrival process with rate of 0.32 and their respective costs could be approximated by the distribution $\log_{10}N(7.20, 0.67)$. Because JIEDDO did not establish a quantitative way of measuring initiate value (see previous chapter), we are left to speculate what values a rigorous method might produce.

In order to ensure that the DSKP Recursion algorithm operates over a broad range of values – so that it might adequately populate the array of critical values – we chose to characterize the distribution of initiatives as Uniform[0, $2c$]. Thus the distribution of value has a mean equal to the cost of the arrival and ensuring that, on average, value is proportional to cost. We impose this requirement not because its believed to be true, but because a) nothing is known about the distribution of value, and b) to ensure that the DSKP DP algorithm is able to adequately evaluate values over a sufficient range to populate the $R_t^b(c)$ array.

Curse of Dimensionality. One of the challenges for DP is the "Curse of Dimensionality". The WPSP is a good example of this problem. Recall that we use "value" as length of the arcs in our network. Budget reflects the state of the system, be it dollars, thousands of dollars or millions of dollars. The problem with WPSP is that costs are log normal: we found costs of individual decisions spanning more than 4 orders of magnitude. This could lead to network with 10,000 states, 50 stages, (given the long right tail of the log-normal distribution) 2,000 bins to discretize the cost distribution, and 50 value bins per cost bin. The resulting network would have, worst-case, 500,000 nodes and on the order of 50 billion arcs. The example which we will use has 1,200 nodes and 561,600 arcs. DSKP DP run time is proportional to the number of arcs, so we should expect the full problem to take at least 50,000 more cycles to run.

Example Problem Data. We will work with a smaller scale numerical example:

T = 12 time periods

B = 100 resource units

p = 1/3 - likelihood of binary arrival in any given time step

$C \sim$ Log-normal(2, 0.5) distribution of arrival cost, (discretized over 234 bins)

$V \sim$ Uniform(0, 2C) distribution of arrival value given its cost (20 bins)

DP and ADP Computational Details: All DP and ADP algorithms were coded in R and run on an Apple Macbook Pro with 2.3 GHz quad-core processor.

Results. First, lets develop some intuition on the log-normal distribution. The histogram below shows a histogram of 5,000,000 samples from the distribution *C*.

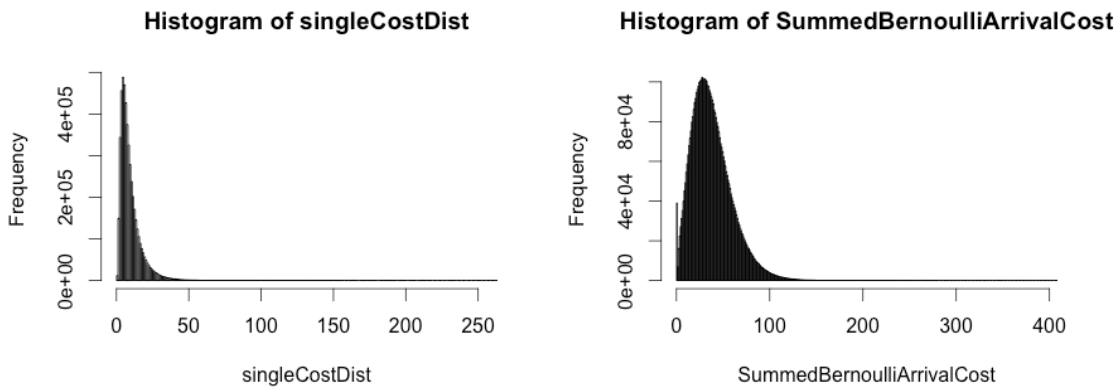

Figure 4-12 Histograms for 5M replications of the cost of a single arrival and for the total cost of all arrivals

We can calculate the mean of the cost distribution for a single arrival via the formula $e^{(\mu+\frac{\sigma^2}{2})}$ which, for the given data, equals 9.483. In twelve time-periods with probability of arrival of 1/3, we expect four arrivals and a total cost of 37.930 – well under the total budget of 100 units. Interestingly, we know that the sum of log-normal random variables is also log-normal but there is no analytical solution for the parameters of the distribution.

The long right tail tells us that there exists a small but real likelihood that a single arrival could exceed the budget: 1.17e-10. For the 12 time-step epoch, the probability that the resource budget would be exceeded is estimated via numerical integration at 0.012.

Running the model, the first views shown are marginal plots showing how the optimal cumulative value *f*() varies as with stage *t*. Given the slow arrival rate, clearly more time would be needed before the budget was effectively used.

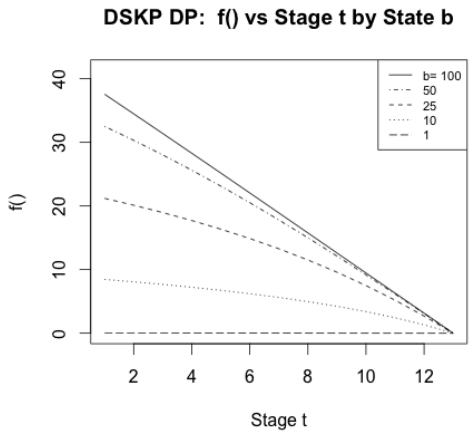

Figure 4-13 Cumulative Value vs Stage t

This figure shows that at the full budget of 100 units, the system experiences essentially linear growth – essentially every arrival is accepted. As the starting budget decreases the long heavy tail of the cost distributions begins to make itself felt, with a

growing likelihood that one of the four expected arrivals in the epoch will have a cost that consumes most of the available budget. With more time prior to the deadline and a smaller budget the decision to accept is more conservative. With a small budget it becomes difficult to accrue value since only low cost initiatives can be accepted and these have a low likelihood of arriving. The next figure depicts cumulative optimal value *f*() vs. resource state b, which is the same surface in the previous figure but rotated about the left vertical axis.

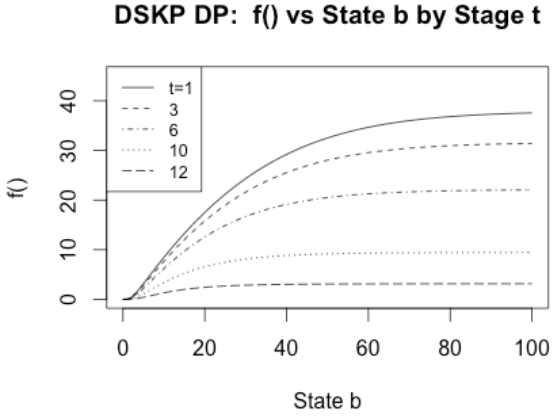

Figure 4-14 Cumulative Value vs State b

Note that for a given state t the curve is not strictly convex: from b=0 to b=1 there is little growth in value and then the slope increases rapidly before leveling off with diminishing returns. For the log-normal distributions small values are unlikely and so

when the budget is small, an affordable arrival is unlikely. This lack of monotonicity is an example of what Papastavrou, Rajapopalan, and Kleywegt (1996) call inconsistent behavior, a result of the cost distribution not being concave and monotonically decreasing.

The concern is that distributions that are not concave and monotonically decreasing can lead to degenerate behavior. The figure below, taken from their paper, compares consistent behavior on the left with very degenerate behavior on the right.

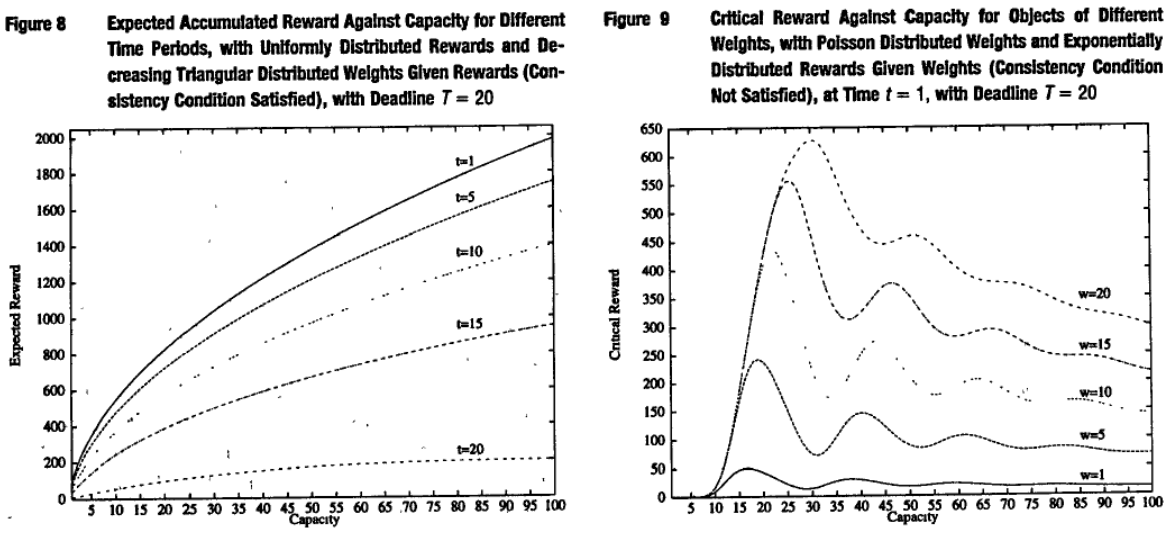

Figure 4-15 Excerpt from Papastavrou, Rajapopalan, and Kleywegt comparing Consistent and Inconsistent Behaviors

The degenerate behavior shown above on the right for critical reward, provides strange results. For example, for an item of weight 15 and critical reward 350, the policy would recommend that one procure the good at capacity 35 and 55 but no at capacity 45.

Looking back at Figure 4-14 we can see that except for that small region of concavity at very low budget levels, the curve behaves consistently. The authors' definition of consistency required a cost density distribution that was monotonically decreasing throughout its domain. A couple example distributions they mention are the right triangular distribution and the exponential distribution. The log-normal distribution does not follow this behavior but rather, it has an "early" period of increasing value followed by a long monotonically decreasing tail. The figure below compares the log-normal distribution to the exponential for similar means.

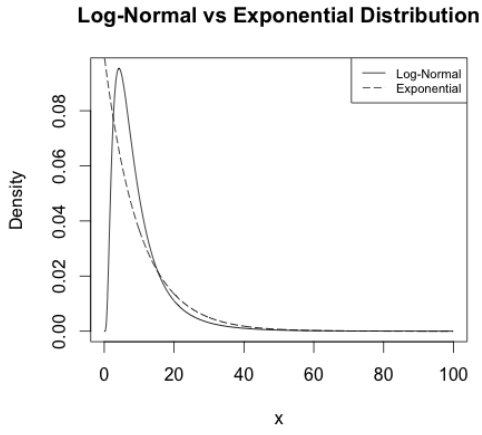

Figure 4-16 Comparing the Log-Normal and Exponential Distributions for the Same Means

We conclude from these examples that other than when the problem space where the budget is small – which we will loosely define as when the resource state is smaller than expected cost of a single arrival -- we can expect consistent behavior. The figure below shows the Critical Reward thresholds for a selection of costs at stage 6 – the halfway point of the decision epoch. The Critical Reward is what will be used as a decision aid: for an arrival at stage t, with cost c and value v, with a given budget of b, we accept the item if v is greater than $R_t^b(c)$.

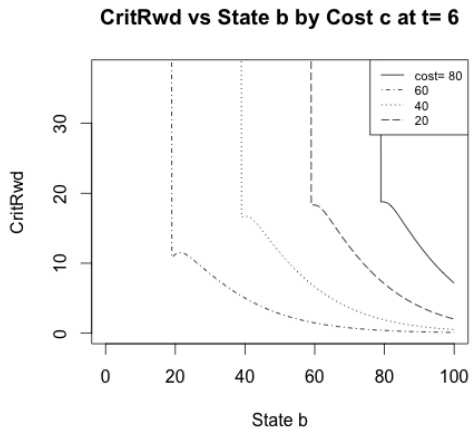

Figure 4-17 Plot of Critical Reward vs Resource State by Cost of Arrival at Stage t=6

First we note the vertical lines on the curves. This is graphical depiction of the constraint that the cost of the item must be less than or equal to the resource state b. Then

observe the "notch" in each curve. This is another example of the inconsistent effect. Connecting the notches provides the curve of $f(b, t=6)$ that was depicted in Figure 4-14. Surprisingly, the sum of log-normal variables is not well understood – no closed form exists - but has been approximated successfully via a single log-normal distribution with several recipes for specifying the parameters (Wiu et al, 2005). This provides insight on the shape of the critical value curves. At t=6, the expectation is for two initiatives to arrive prior to the deadline, so the distribution of the total cost should follow the distribution of the sum of two identically distributed log-normal random variables, which in shape should follow a log-normal distribution. Since the distribution of initiative value is directly proportional to cost, it follows that the distribution of critical reward as a function of resource state *b* given initiative cost appears to follow a log-normal distribution as well.

Curse of Dimensionality Revisited. This small numerical example requires approximately one minute of run time as implemented in the statistical package R on a 2.3 GHz MacBook Pro. Since the full problem is approximately 50,000 times larger, we should expect a similar computer, without taking any steps reduce the dimensionality of the problem, to require about 35 days of run-time to solve the full problem.

Comparing the Bernoulli Arrival Process as an Approximation to the Poisson. As described in the introduction to this paper, the number of arrivals for a given period was assumed to follow a Poisson distribution. This poses a problem for either of

the approaches we have chosen, since it means the number of the arrivals is theoretically unbounded.

From Papastavrou et al (1996) we use the approach of discretizing time to a unit step small enough where the likelihood of more than one arrival is sufficiently small. We then use a Bernoulli arrival process as an approximation of the Poisson. The natural question then is how close is the Bernoulli approximation? Employing a probability of arrival of 1/3, the Bernoulli arrival process yields an arrival process that follows the well-known binomial distribution: the number of successes x in n Bernoulli trails where each trial has the same probability of success p. By the Poisson Limit Theorem, the Poisson distribution converges to the Binomial for large n and small p. We compare first the number of arrivals.

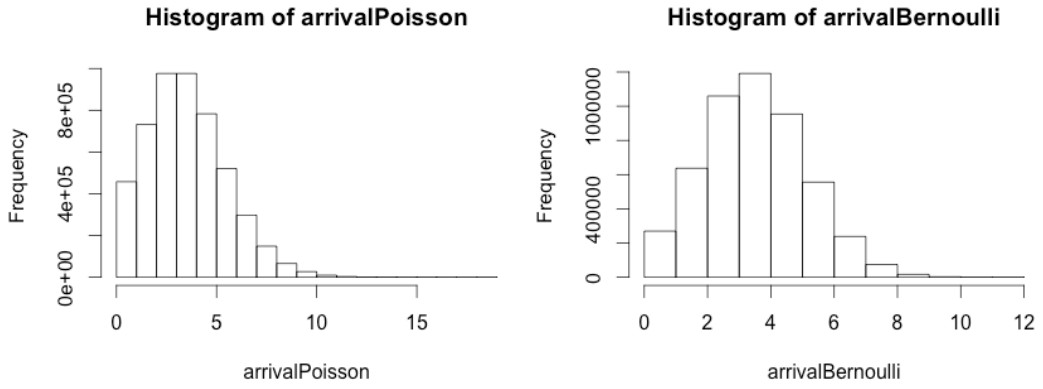

Figure 4-18 Histograms comparing Poison and Bernoulli arrivals in a discrete time period of 12 steps with mean arrivals per time period of 1/3

Clearly, 12 is not a large enough n and 1/3 not a small enough p. The next comparison examines the effect of the approximation on the cumulative cost distribution.

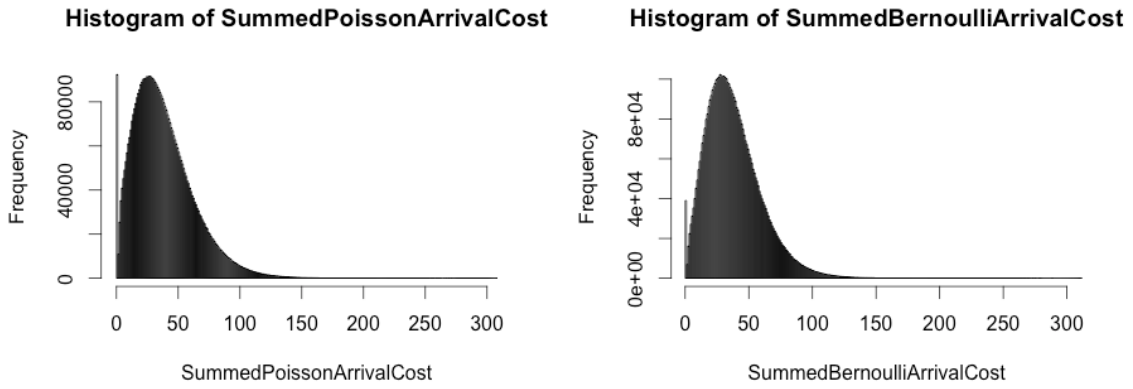

Figure 4-19 Histograms comparing Poisson and Bernoulli arrival processes total cost distributions

The resulting difference is nuanced. The Poisson shows a stronger mode at zero while the Bernoulli arrival process has a weaker mode. This is on the whole due to smaller variance of the Binomial compared to the Poisson for these data. Practically speaking the difference is small. Numerically integrating shows that the likelihood the Bernoulli arrival process results in exceeding the budget of 100 units is 0.012 while for the Poisson the likelihood is 0.020. On the whole this approximation, for a small price in accuracy, provides us with a viable way of modeling the problem by bounding the number of arrivals.

Insights. While using the DSKP DP algorithm to solve real-time instances of the WPSP is not practical, it does provide valuable insights. The big insight is that the optimal rate of value accrual is super-linear – meaning that it does not just follow a linear rate. It is optimal to be more conservative at the start of the period and then be more aggressive as time runs out. The results are not precisely scalable: there is a complex relationship between the arrival rate, the distribution of cost, and the duration of the decision epoch. Basic results can lead to simple observations: items with a value to cost ratio less than the linear rate are never optimal to acquire.

Acquisition Phases. A challenge to realistic implementation of the WPSP is that the problem is not as myopic as we have modeled it. Commonly, there is some knowledge of arriving items, some decisions can be deferred at a cost, development can be slowed down or sped up, and development is done in phases. Never is an initial decision made with all the resources required to deliver that value fully committed. Rather, items are given funding increments over developmental phases with intermediate goals or gates that must be achieved. This limits risk without slowing the process down untowardly. These subsequent decision points can be and usually are scheduled, in contrast to the random arrival process. Thus, at a minimum the WPSP requires that we capture the notion that at every stage t we have the possibility of arrivals and also the possibility of scheduled decisions. This added complication on top of the previously discussed curse of dimensionality drives home the reality that if we are to use DP to

address this problem, we need a different approach: Approximate Dynamic Programming.

4.4 Approximate Dynamic Programming Implementation of WPSP

Approximate Dynamic Programming (ADP) is a relatively recent approach. When interpreting the shortest path problem in the case of random travel times over arcs, one way to think about is to envision navigating *forward* through the network repeatedly and learning the shortest path. For well-solved problems, it is not the ideal technique. Exact methods produce better answers faster. According to Powell (2007) it is at its best when attempting to solve sequential decision problems that are unsolvable via exact methods because of being large, or where the state transition function are not known.

ADP has a variety of approaches for handling different kind of problems, but one of the major categorizations is based on whether problems have finite or infinite horizons. Our focus is on the finite horizon problem: the decision maker has a budget that she must use or lose by the end of the decision epoch.

Recall that in stochastic DP, $f()$ is an expectation. In ADP, we make repeated trips through the network, where each trip generates a sample of cumulative value by node. Powell in his notation has this sample of shortest path values as v_t^n and the expectation is V_t^n. To stay consistent with our notation up till now, we will make the n-th sample f_t^n, leaving it un-italicized to differentiate from $f_t^n()$, the expectation after the n-th iteration. We sketch a simple algorithm of the approach.

Simple Finite Horizon ADP WPSP Algorithm

Step 0. Initialize $f_t^0(s)$, set iteration $n = 1$, set b.

Step 1. Generate a sample set of arrivals over the decision epoch.

Step 2. For t = 1 to T {

If there is no arrival let $c_t = 0$ and $v_t = 0$

Solve the Bellman equation:

$$f_t^n = \max_x\{v_t x_t + f_{t+1}^{n-1}(b_t - c_t x_t)\}, \text{ s.t. } c_t <= b_t$$

Update the expectation, where k is the counter for the visits to node (t,b_t):

$$f_t^n(b_t) = ((k-1)/k)^* f_{t+1}^{n-1}(b_t) + (1/k)^* f_t^n$$

Update the state:

$$b_{t+1} = b_t - c_t x_t$$

}

Step 3. While $n < N$, increment n. Return to Step 1.

Step 4. Return $f_t^N(b_t)$ for all t and b.

This algorithm is what Powell calls the single pass procedure. He also describes the double pass procedure, where the difference lies in how the updating of the expectation is conducted. In the single pass, the expectation is updated at every stage while making the trip through the network. In the double pass, we find the shortest path through the sample network, keeping track of the nodes visited and the decisions made. We then travel backwards, and use that shortest path information to update the

expectation at each node that was visited. The difference is subtle but significant since the single pass could suppress extreme behavior.

Double Pass Finite Horizon ADP WPSP Algorithm

Step 0. Initialize $f_t^0(s)$, set iteration $n = 1$, set b.

Step 1. Generate a sample set of arrivals over the decision epoch.

Step 2. For $t = 1$ to T {

If there is no arrival let $c_t = 0$ and $v_t = 0$

Solve the Bellman equation:

$$x^t = \text{argmax}\{v_t x_t + f_{t+1}^{n-1}(b_t - c_t x_t)\}$$

Update the state:

$$b_{t+1} = b_t - c_t x_t$$

}

Step 3. For $t = T$ to 1 {

Let $b_{T+1} = 0$

Compute shortest path distance to t

$$f_t^n = v_t x_t + f_{t+1}^{n-1}$$

Update the state:

$$b_t = b_{t+1} + c_t x_t$$

Update the expectation:

$$f_t^n(b_t) = ((n-1)/n)^* f_{t+1}^{n-1}(b_t) + (1/n)^* f_t^n$$

}

Step 4. While $n < N$, increment n. Return to Step 1.

Step 5. Return $f_t^N(b_t)$ for all t and b.

For the WPSP this approach gives us a better picture of what a given decision epoch might look like, since the expectation is updated using the entire shortest sample path versus updating the expectation one arc at a time.

Previously we used the array of Critical Reward as the output that would be used to support decision making, which is not an output here. We can still use this concept but we need not calculate the entire array. For an arrival at stage t for resource level b with cost c, we can readily calculate the Critical Reward using its definition.

Equation 4-17: ADP Critical Reward

$$R_t^b(c) = f_{t+1}{}^N(b_t) - f_{t+1}{}^N(b_t - c_t)$$

4.4.1 Initial Implementation

Pre-decision and Post-decision state variables. Previously we discussed how the state space could be partitioned to differentiate between when the system is affected by decisions and when it is affected by exogenous effects. This is an important concept in ADP, particularly in addressing Markov Decision Processes, where the next state of the system is conditioned both on the decision and on exogenous effects. However, in

the WPSP, the exogenous effect is the arrival and its accompanying descriptive data. Once a decision is made, the state transition is deterministic: the resource state (budget) is decremented accordingly and the system advances to the next stage. Thus, we initially used the pre-decision state variable to avoid the unnecessary complication. As we will see, the post-decision state variable proved vital later when we needed to approximate the value function.

"Look-up Table" ADP Results. Since the value function is estimated by keeping a look-up table of both the value estimate and the number of visits, to differentiate this from later implementations we call this version the "Look-up Table" ADP. Running this algorithm in order to generate smooth information requires varying the starting points – both for stage and the resource state variable – since certain trajectories are more likely than others and thus the states in these trajectories will get more visits. Varying the starting points helps fill out the value space. Despite this, the results can be noisy, as depicted in the figure below.

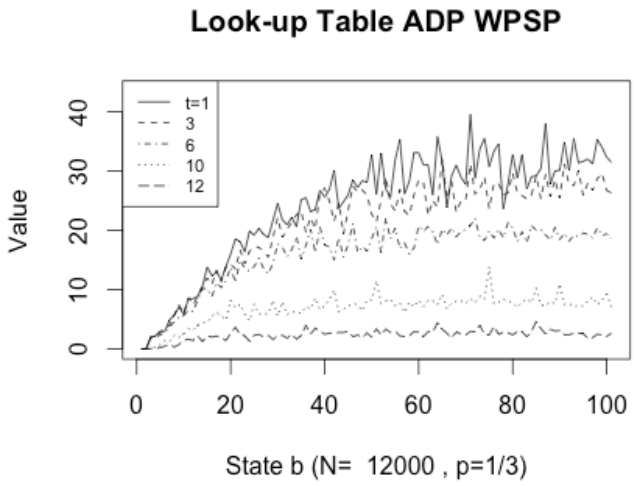

Figure 4-20 ADP WPSP Results for 12,000 Iterations

To smooth these results, one technique employed was to run an outer loop algorithm that incrementally increased the default starting budget, although within each outer-loop iteration we still varied the starting point within the default starting budget. For each outer-loop budget value, the final cumulative value at a selected set of stages was recorded. This resulted in a much smoother graph at the price of 34 outer loop evolutions, each of 12,000 iterations.

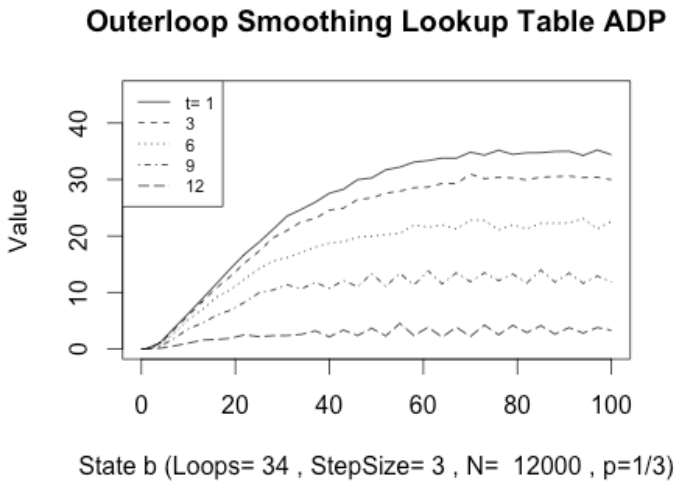

Figure 4-21 ADP WPSP Results: Smoothed Via Outer loop Procedure

While we have shown that the ADP implementation can replicate the results from the DP DSKP algorithm, the approach, at this point and for the given set of data, is far less efficient. Fundamentally, we must visit every state multiple times in order estimate the value function. The power of ADP lies in being able to maximize the value of the information gained from every trip through the network, exploiting problem structure and statistical methods to approximate the value function, versus trying to directly calculate the sample mean state by state.

4.4.2 Approximating the Value Function

For this problem, the power in ADP lies in exploiting regression to approximate the value function. The idea is to explore the space and "learn" the value function as an analytical function of the state space. Only via an approximation will we be able to run large-scale problems – cases of high variance and large orders of magnitude. In order to do value approximation we will need to employ three concepts that are central to ADP: learning, the post-decision variable, and recursive regression.

Learning and the Stochastic Gradient Function

In our initial ADP implementation, we kept track of how many times we visited each node in order to estimate the value function at that node. This a modification on the basic updating equation:

Equation 4-18: Basic Value Function Updating Equation

$$f_t^n(b_t) = \frac{n-1}{n} f_t^{n-1}(b_t) + \frac{1}{n} \mathrm{f}_t^n$$

This a recursive equation for calculating the sample mean.

Equation 4-19: Sample Mean Equation

$$f_t^n(b_t) = \frac{1}{n}\sum_{m=1}^{n} \mathrm{f}_t^m$$

Powell tells us that what is nice about using Equation 4-18 is that it is very simple and easy to calculate. The problem with this approach in practice is that it places excessive weight on the information obtained early in the exploration of the space and, because it has a tendency to approach zero quickly, it provides too little weight to the information gained later when the values are becoming better established. We modified the estimation in the initial implementation to keep track of how many times each state was visited since the odds of visiting any single state are slim. However, this approach only made the overall approach more cumbersome computationally.

Powell approaches this issue by starting with the basic theory of how to approximate the mean algorithmically, using the stochastic gradient algorithm. Assume that we have a vector of observations R from a random process W. We wish to find the number θ that minimizes the mean squared error. We define the error function $F(\theta, W)$.

Equation 4-20: ADP Error Function

$$\min_{\theta} \mathbb{E}\{F(\theta, W\} = \min_{\theta} \mathbb{E}\{\frac{1}{2}(\theta - R)^2\}$$

Taking the derivative and setting it equal to zero would be the standard solution. However, let us further assume that the observations come from a random process and that we can only see one observation at a time and we need to update our estimate as we go. Let $R(\omega)$ be the current sample. Then we can consider the sample error function.

Equation 4-21: ADP Sample Error Function

$$F(\theta, \omega) = \frac{1}{2}(\theta - R(\omega))^2$$

The gradient of this function is just the difference between θ and $R(\omega)$. It is stochastic since it depends upon the sample from a stochastic process. This gradient allows us to recast the problem of updating the estimate with each sample as a kind of search problem. Nash and Sofer present a basic search equation that consists of the previous solution, an improving direction, and a step size. Powell embeds this gradient in the standard search equation, which uses the last estimate θ^{n-1}, the stochastic gradient, and a step size α_{n-1} to yield the next estimate θ^n.

Equation 4-22: ADP Stochastic Gradient Equation

$$\theta^n = \theta^{n-1} - \alpha_{n-1}(\theta^{n-1} - R(\omega^n))$$

We turn our attention first to the issue of step-size, α_{n-1}.

Step Sizes

Powell studies in detail the question of the step size. As discussed earlier, in this type of stochastic search using 1/*n* will lead to an artificially early convergence. For most problems, Powell tells us, there is not an optimal search step. The challenge is that we are trying to estimate the value function one random sample at a time while systematically searching over a policy space. He offers many strategies that are each best suited to particular ADP applications. Often strategies can be used in combination.

One simple strategy to prevent premature convergence, and one that is suited to approaches involving many parameters, is to use a constant step size. This is particularly useful when we wish the algorithm to have ample opportunity to explore the policy space. It has the disadvantage that the step size remains constant rather than allowing it to decay to zero.

For our application, based on exploration with a variety of rules, we altered the concept of constant size. We allowed the algorithm to search with a constant size for a

proportion γ of the total runs. At every iteration we applied a multiplicative factor $\beta < 1$ to α so that it would exponentially decay to zero. This strategy is shown below.

Search-Then-Decay Step-size Strategy

$\boldsymbol{\alpha_n = \alpha_0}$, if $n < \gamma N$

$\boldsymbol{\alpha_n = \beta \alpha_{n-1}}$, if $n \geq \gamma N$

Setting the values of α_0, γ, and β required trial and error. The intent is to allow the algorithm to spend most of the time searching. Once values have stabilized we then allow the step-size to decay towards zero. One way to do this is to track the sum squared error (SSE) – the square of the stochastic gradient. If the SSE becomes stable, this is an indicator that the step-size is allowed to decay. Once the decay point γN is reached, the aim is to set β to a value that bring the step-size down close to zero in $N(1 - \gamma)$ remaining iterations.

As to the fixed step-size α_0, Powell discusses how special care is required when selecting fixed step sizes for recursive regression models using basis functions.

Post-Decision State Variable

When we kept a look-up table of the value function, the post-decision state variable seemed an unnecessary complication. However, it is vital for a value

approximation scheme. Fundamentally, we want to regress on the post-decision state because this significantly reduces the variance in the estimation problem. If we were to regress on the pre-decision state variable, our regression model would have to account for both the variance in the random process and the variance in the decisions induced by the systemic search. Regressing on the post-decision state variable removes this decision-induced variance.

Powell indicates the post-decision state variable by appending the superscript x to the state. Conceptually, it also easy to understand: for a given budget b and an arrival with cost c, our point of comparison is the value of passing on the buying opportunity and staying at $b^x = b$; or acquiring and moving to state $b^x = b$-c.

Recursive Regression

Regression has the attractive property that we can avoid the requirement of using a continuous state variable – the amount of resources – as both a discrete variable and also as an index to the state space. Instead we can use the resource state as an independent variable and use it to compute an approximation to the value function.

The standard regression equation found in most textbooks is shown below.

Equation 4-23: Standard Regression Equation – Vector Form

$$y = \theta^T x + \epsilon$$

Here we have used a vector form of the equation. If the original decision variable x has I dimensions, we have added an additional element $x_0 = 1$ for the intercept term θ_0. We note, however, that our value function is clearly not linear and we only a single dimensional predictor b. As we will see, we will have a different model for each stage t. What Powell stresses is the need for basis functions, so that the approximation function is now a function of both the state b and the parameter vector Θ, via a set of basis functions ϕ_j. This makes our approximate value function as follows.

Equation 4-24: Regression-Based Approximate Value Function

$$\overline{f}_t(b_t^x, \theta) = \sum_j \theta_j \phi_j(b_t^x)$$

Note that we are using the post-decision variable. The challenge now is how to transform this equation into a recursive approach. The answer is to embed this equation into the stochastic gradient search equation.

Equation 4-25: Recursive Regression-Based Approximate Value Function

$$\theta^n = \theta^{n-1} - \alpha_{n-1}(\bar{f}_t^{n-1}(b_t^x, \theta) - \mathrm{f}_t^n)\nabla_\theta \bar{f}_t^{n-1}(b_t^x, \theta)$$

The gradient with respect to θ is the vector of basis equations.

Equation 4-26: Recursive Regression-Based Approximate Value Function Gradient Term

$$\nabla_\theta \bar{f}_t^{n-1}(b_t^x, \theta) = \Phi(b_t^x)$$

Revisiting the step-size rule previously described, Powell states that in recursive regression there is frequently a scaling problem involved. He recommends adjusting the initial step-size so that initial changes in θ_j run 20 to 50 percent, in order to assure that the algorithm do not allow the values to swing wildly and fail to converge. This was the case for this problem initially. We finally settled on an initial alpha of 0.01. While this may seem ridiculously small, it proved very effective. The figure below shows a smoothed plot of SSE during a run using this initial step-size in the Search-then-Decay strategy. Despite the small step size, SSE values are still large and variable, from the log-normal distribution heavy tails.

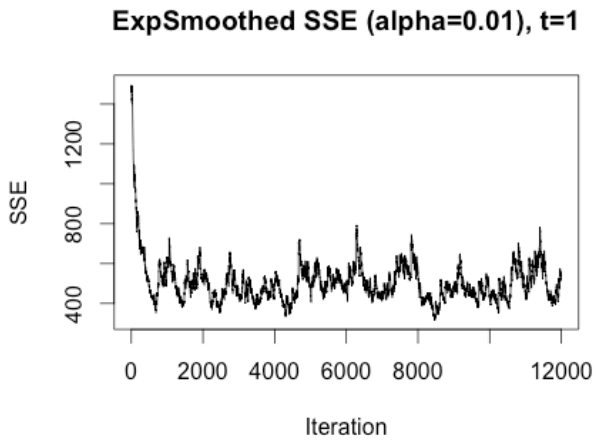

Figure 4-22 Smoothed Plot SSE for Stage t=1

The issue then becomes how to choose basis functions. Konedaris (2008) describes many approaches to value function approximation basis functions, and emphasizing the utility of the Fourier series. These and other sigmoid basis functions such as the Gompertz curve were tried. While none of these worked particularly well, this exploration led to the insight that the value function is ultimately a weighted sum of the value distributions of the arriving initiatives. Since these are uniformly distributed and proportional to the cost distribution, then the value function should have the shape of a log-normal cumulative distribution function (cdf). The question was: what parameters to use? The parameters for the log-normal distribution have to remain constant, since the search algorithm varies the regression parameters, not the basis functions themselves.

We conducted some preliminary investigation of using log-normal cdf basis functions for linear regression using output data from previous runs. It became apparent that for different stages, the best fit required a different set of log-normal parameters.

The approach we chose was to use a set of log-normal basis functions with the means drawn from a sequence that started with the means of a single arrival and proceeded up in steps to a mean larger than might be expected to arrive during the entire decision epoch. For the set we choose to use a log-variance of 1.0 since from our preliminary investigation this seemed to be a good choice. The idea was that for a single time step we should get exactly the mean of a single arrival, save for the randomness of the arrival process while, for the full epoch, we would get a log-normal with a mean close if not equal to sum of all the arrivals.

For our initial attempt via this example problem we used 6 basis functions, with the log-means sequence running from 2.0 to 4.0, in steps of 0.4. The figure below shows the resulting approximated value function alongside our initial attempts.

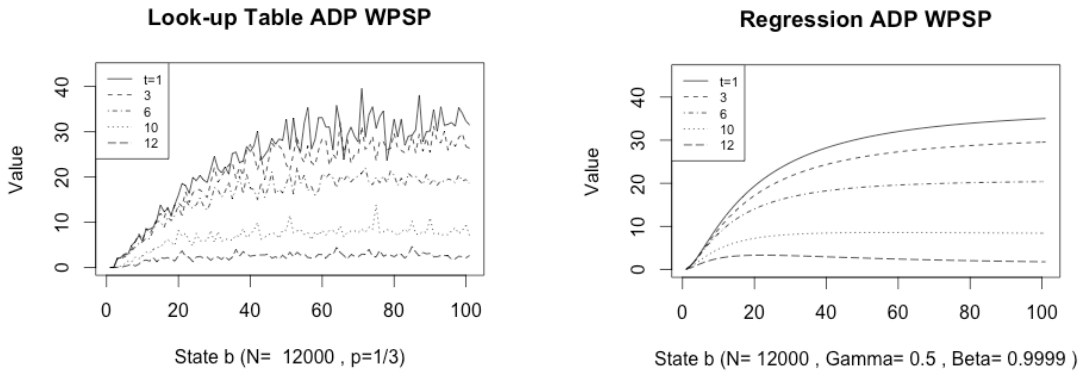

Figure 4-23 Side by Side Comparison of the Table Look Up Value Function vs Recursive Regression Value Function Approximation using Log Normal basis functions

Qualitatively, there seem to be two areas of concern: for late stages (t=12) the slope actually goes negative and at lower values of b the slope appears too steep compared to the output. Other than this, the value approximation seems to compare well with the table look-up approach. It also begs comparison to the "exact" values obtained from the DSKP DP algorithm. The figure below compares these two results.

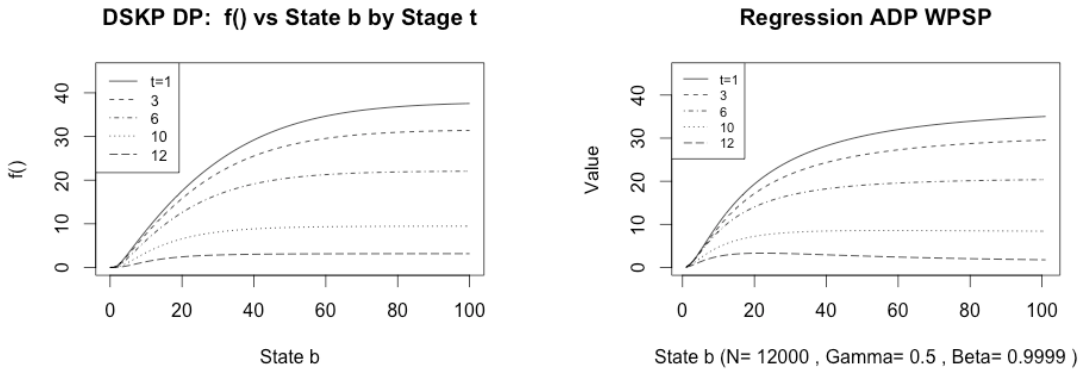

Figure 4-24 Side by Side Comparison of the exact DSKP DP solution vs Recursive Regression Value Function Approximation using Log Normal basis functions

This comparison reinforces the previously observed negative slope in the late stages and subtle differences in the shape of the curve. We also note that there is too little concavity for very small values of *b*. It is important to get this fitting on a sound basis because for the full-scale problem we will not have the luxury of an exact solution for comparison.

This led us to question of whether there was a "right" number of basis functions. In order to explore the right number we needed to understand how to choose the sequence of log-means for a given number of functions. In the previous sequence, we used a sequence from 2 to 4. To make the process more general, we have the range for the sequence start on the mean (not log-mean) of the distribution, which in log space is $\mu + \sigma^2/2$. The upper range of the sequence we set as the product of $\exp(\mu + \sigma^2/2)$ and the number of expected arrivals over the entire epoch. Translating this back into log space

the upper extreme of the basis range is $\ln(pT\exp(\mu + \sigma^2/2))$. The approach we chose was to use this range to generate a basis sequence would be to use mid-points of increasing numbers of even partitions. The table below illustrates this process for the one through four bases.

Table 4-4 Selecting Log-Means Based on Number of Basis Functions

# Bases	LL	Log-Means				UL
1	2.25		2.94			3.64
2	2.25		2.71	3.17		3.64
3	2.25	2.60	2.94	3.29		3.64
4	2.25	2.53	2.80	3.08	3.36	3.64

Using log-means generating via this sequence, the figure below shows a panel of the same plot of the Regression ADP for the set of bases {1, 2, 3, 4, 6, 12}.

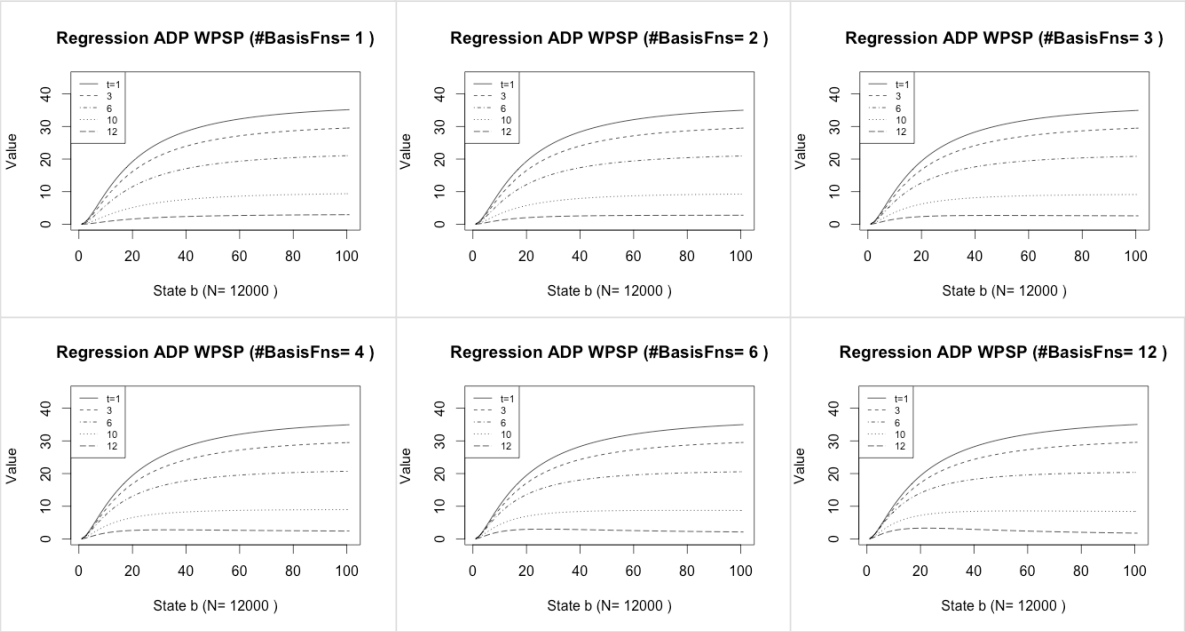

Figure 4-25 Panel of Recursive Regression Value Functions Approximations Varying the Number of Basis Functions with log variance of 1.0

To compare these numerically, we examine the mean of the smoothed sum of squared errors (MSSSE). As shown in Figure 4-22 we use an exponential smoothed sum of square errors to observe convergence behavior. Observing the mean of this time series provides a useful basis for comparison among different numbers of basis functions. To ensure that these comparisons are useful, we reduce variance by using the same random seed throughout.

As the table shows, these are very close. Perhaps only a few basis functions are really needed.

Table 4-5 Mean Smoothed Sum of Squared Errors for Different Number of Lognormal Basis Functions

# Bases	MSSSE
1	250.73
2	249.98
3	249.97
4	250.24
6	251.18
12	254.96

Even a single log-normal basis function captures the basic dynamics. Comparing the shapes of the curves to the DSKP DP solution, there is still a relative lack of concavity with too sleep for t=1. The issue seems to be that in this regression scheme the "wrong" basis functions are still allowed to have some influence over the fit. For example, the log-mean of 2.25 equates the mean value we expect for a single arrival – which is a scenario aligned with t=12, the last stage. Yet a 12-basis function scheme would allow this particular basis to have influence over stage 1, from which we expect four arrivals. Even a scheme with only two basis functions can produce negative coefficients for some the stage solutions as per the table below.

Table 4-6 Regression Coefficients Resulting from a Two Basis Function Regression Solutions to the ADP WPSP Example Problem

BasisLogMean	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10	t=11	t=12
2.17	19.16	21.41	22.64	24.54	25.67	26.61	25.52	25.30	23.08	20.72	17.53	12.39
3.17	18.23	12.85	8.33	3.29	-1.01	-5.21	-7.26	-10.08	-11.16	-12.21	-12.33	-10.58

The other aspect that required examination was the variance of the basis functions. The various basis functions used above all employed the same log variance of 1.0. We arrived at this figure from our preliminary regressions on output data; not because it was the ideal figure but, because if only one variance was to be used, it seemed to be the best case *for the data employed*. Given our approach there's no reason we could not employ a set of basis functions where we vary both log-means and log-variance.

We use the same log-means generation approach, but for each log-mean we generate two basis functions. Recall that the distribution of initiative value depends upon the cost c of the most recent arrival, being $U(0, 2c)$. So we should expect that the cdf of the log-normal used to fit should have a log-variance of 0.5 (which corresponds to the cost distribution in this example problem) and one larger – but how much larger?

For the following panel we use 2 and 4 separate log-means, the top row being those with 2 log-means. The three columns compare by size of the second log-variance as a multiple of the first: 50% larger, twice as large, and four times larger.

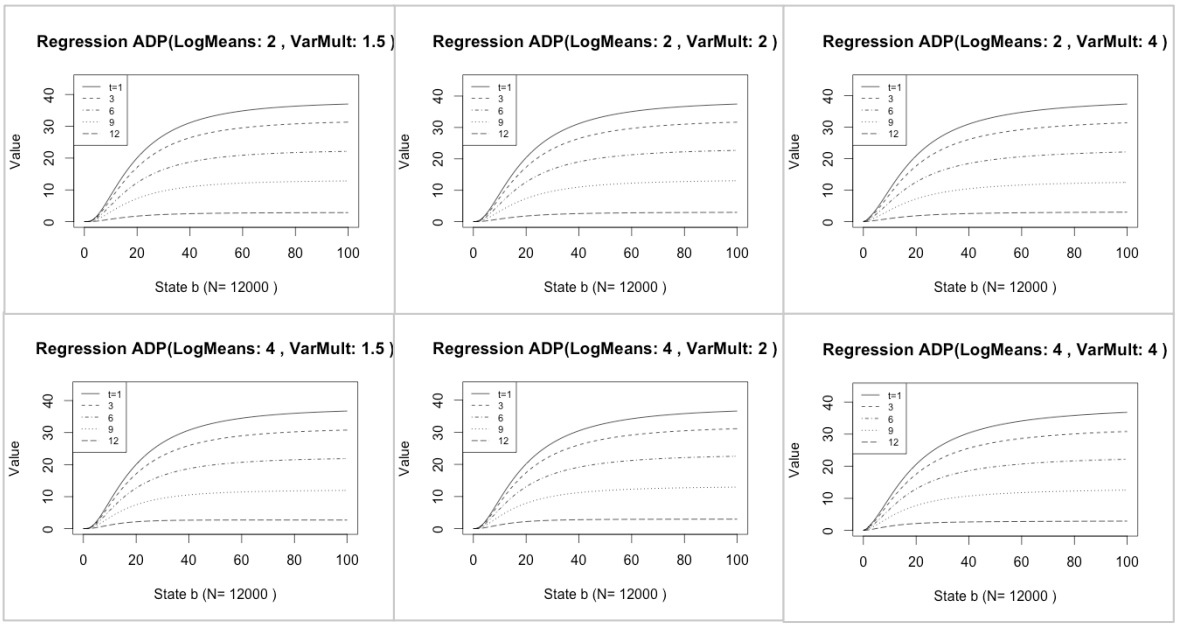

Figure 4-26 Panel of Recursive Regression Value Functions Approximations Varying the number of Log-Means and Log-Variances

As before, we examine the mean smoothed square error versus the number of log-means used. Here the best fit seems to occur when we use four log-means with the variance of the second set being twice the size.

Table 4-7 Mean Smoothed Sum of Squared Errors for Different Numbers of Log-Means with Two Log-Variances

#Log-Means/2dVarMultiple	MSSSE
2 / 1.5	257.12
4 / 1.5	280.49
2 / 2	245.32
4 / 2	244.05
2 / 4	248.39
4 / 4	248.29

Comparing the best of these (per MSSSE) to the DSKP DP allows us to see if matters have improved qualitatively. While it seems that concavity has certainly improved and we have eliminated the issue with negative slope, the general shape of the curve still does not have quite the right shape.

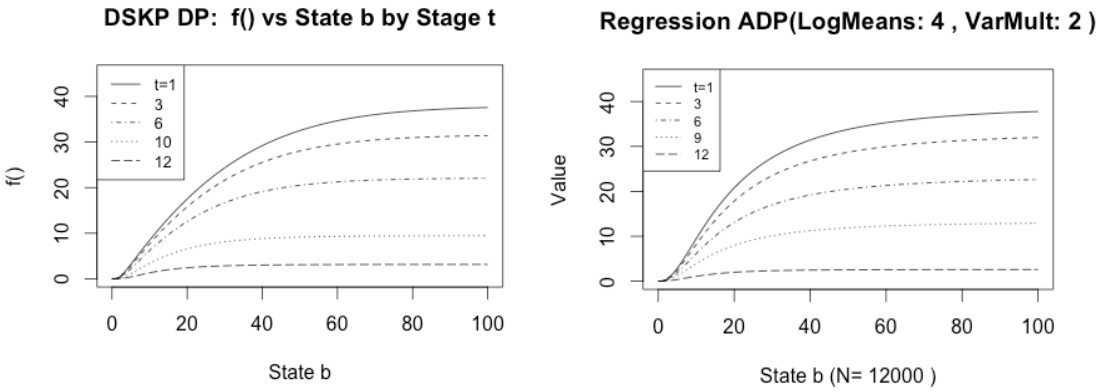

Figure 4-27 Side by Side Comparison of the exact DSKP DP solution vs Recursive Regression Value Function Approximation using eight Log Normal basis functions, using a sequence of four log-means by two log-variances

We decided to explore one more twist on this scheme to see if we could develop a more nuanced regression solution based on the problem's structure. We would have only one basis function, however, we would employ an estimate of the log-mean based upon the number arrivals for a given stage *t*. Thus we would seek to squeeze as much as

possible out of the problem structure. We called this approach the "Tuned Lognormal Bases Regression ADP".

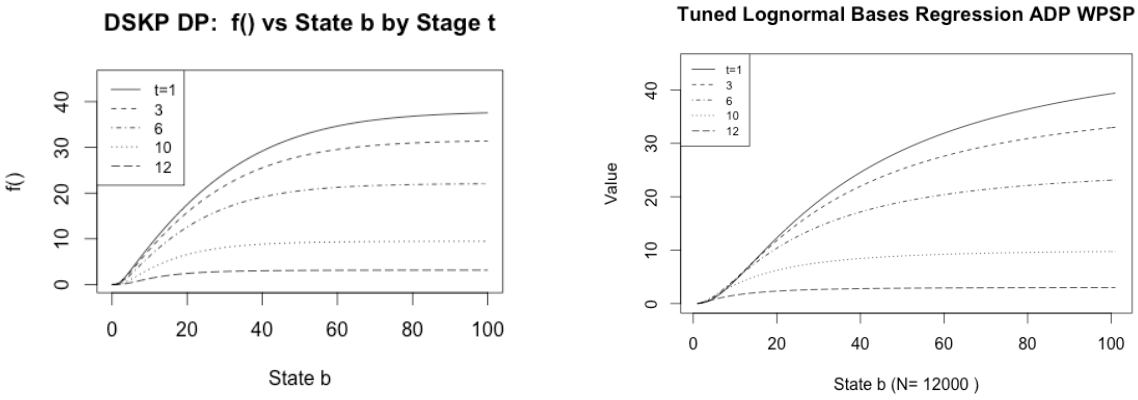

Figure 4-28 Comparing the DSKP Value function to the result from the tuned log-normal bases regression

While the thought was initially attractive, it shows that while the cumulative value functions may have a log-normal shape, that their parameters do not follow neat rules of thumb.

Of the approaches we tried, the approach that seems to have best promise is one that uses a fixed set of bases functions where we combinations of various log means and variances. Our "best of breed" is the case of 4 log-means, using midpoints based on the even partition of a range from the cost of a single arrival to the cost of the expected number of arrivals, and 2 variances, using first the variance of a single arrivals cost and

twice that for the second variance. Our goal is determine the ability of this algorithm to handle the large-scale problem: 9 order magnitude variation in costs and a 50-week epoch. In preparation for this, we briefly consider an important aspect of the problem data: relative resourcing level.

Relative Resourcing Level

We use this term to describe the extent to which the dynamic stochastic knapsack problem is constrained. At one extreme we can consider a severely constrained problem, where the budget is on the order of 25% of the expected total cost of arrivals. For the example scenario this would equate to being able to afford only one of the roughly four expected arrivals. At the other extreme, we consider a budget level that exceeds the cost of all arrivals in the overwhelming number of sampled epochs, a situation which can be described as only mildly constrained.

In the case of the severely constrained budget, we can estimate the resulting cumulative value via a simple "back-of-the-envelope" calculation: the budget equals the expected cost of a single arrival and value is proportional to cost. The latter extreme must be sampled since, while we know the cumulative cost distribution is log-normal, we do not know its parameters. For the middle case, we consider the case where the budget equals the expect cost of arrivals per epoch.

The budgets are, based on our example problem's data, 10, 40, and 100 budget units. We also wish to explore the variance. As we will see later the variance in the full

data set is much larger than in the base case. Thus, we will explore log-variances ranging from a low smaller than the example case to one equivalent to a variance similar to full data set: 0.125, 1.0, and 4.0.

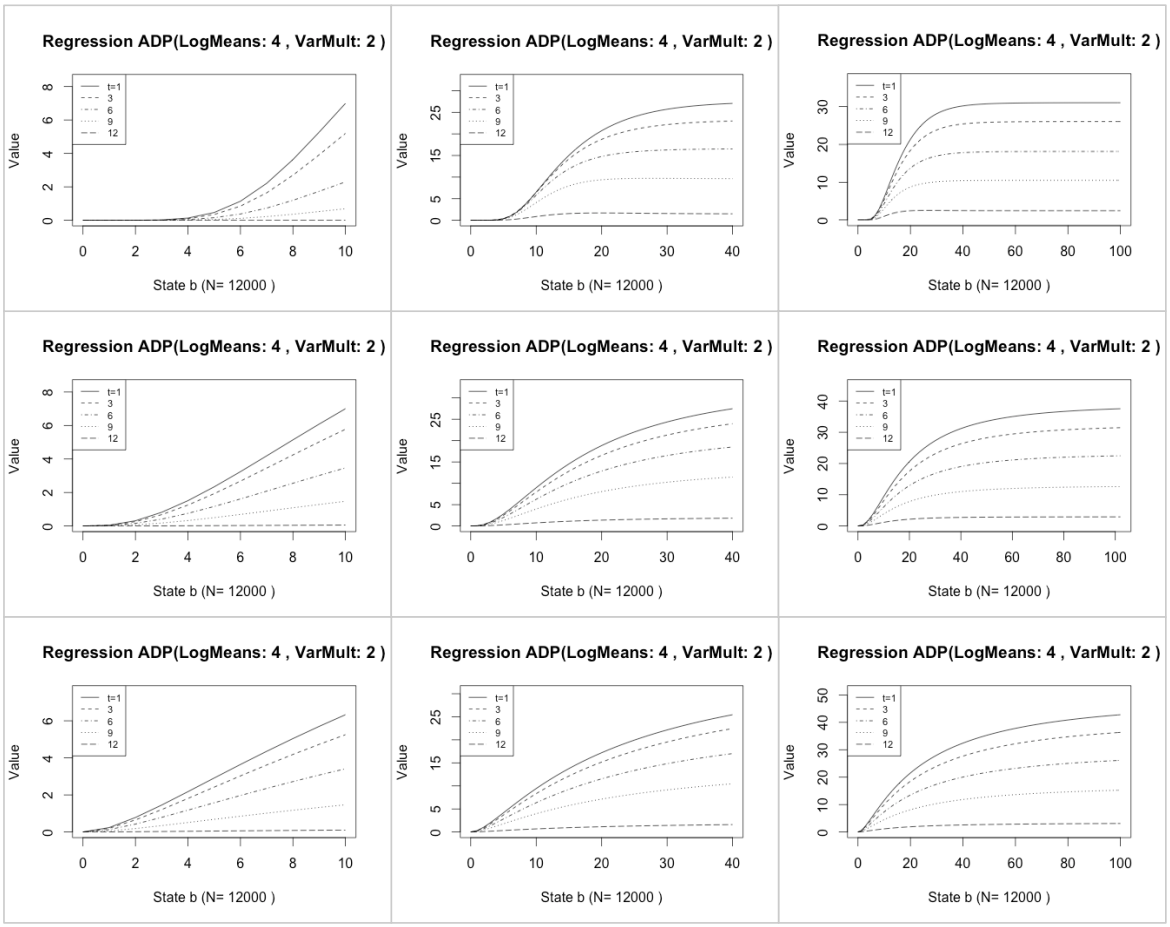

Figure 4-29 Different Constraint Levels, From Severely to Mildly Constrained (from budget of 10 to budget of 100), Compared to Changes in Variance from Low to High (Log-variance = (0.125, 1.0, 4.0))

We can see that because in each case where we have the same log-mean and log-variance, the panels show the same curve, with the only difference being one of scale. With lower variance, budget values much below the single arrival mean allow little to be acquired. At higher variance, the cumulative value line at lower budget levels approaches linearity.

The other tale being told is the contrast between budget, arrival cost, and time in which to acquire arrivals. The left-most column shows the situation where there is just not enough resources. Every additional bit of resources added to the budget produces a strong return. At the other extreme, relative to the amount of value expected in the epoch, there are just too many resources. And, adding resources does not help much – what is needed relative to the resourcing level is to either increase the number of arrivals or to make these of greater value.

Initiative Arrival Rate

The rate at which initiatives arrive is another parameter that impacts the shape of the value curve. The previous examples looked at 12 time steps and a budget of 100 with an arrival probability p of 1/3. What happens as the arrival rate increases?

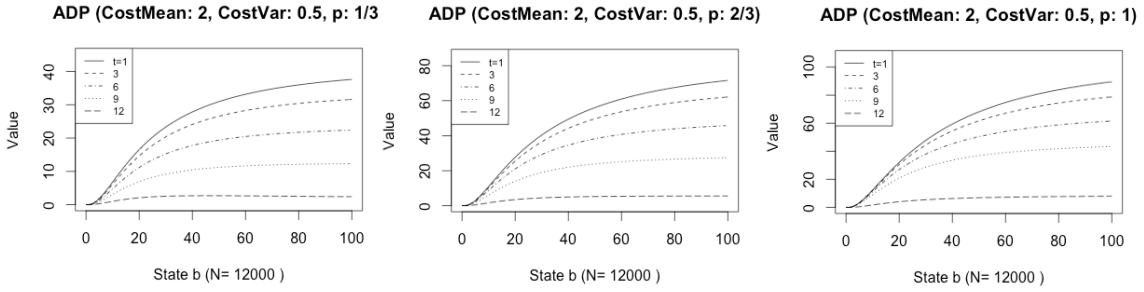

Figure 4-30 Comparison of the Value Curve for the Same Distribution for Different Arrival Rates

One of the observations one might have made earlier was that the amount of value returned relative to the budget seemed deficient. If value was proportional to cost then why was it hard to see this value realized; that is, if the budget was 100 when does the value realized begin to approach 100. The issue was that there was not enough time given the arrival rate; or equivalently, the arrival rate was low given the amount of time. Here we see that the curves are almost identical as the arrival is increased. What has changed is the scale.

Armed with these insights we can test the ability of the Recursive Regression ADP to run the large-scale problem from the JIEDDO case using the data initially described in Chapter 1, and also to make qualitative assessments of the nature of the problem based on its data and resulting cumulative value curve.

4.4.3 Modeling the full scale problem

The full-scale problem employs the following data.

Full Scale Problem Data

T = 52 time periods

B = 2,500,000,000 resource units

p = 1/3 - likelihood of binary arrival in any given time step

$C \sim$ Log-normal(16.58, 3.57) distribution of cost given an arrival

$V \sim$ Uniform(0, 2C) distribution of value of an arrival given its cost

A litmus test is to examine the likelihood that a single arrival exceeds the available budget. For a single arrival we can calculate this directly: 0.0038. While small, this is several orders of magnitude higher than in the example problem. As the histogram shows, this particular distribution has a very long and pronounced right tail.

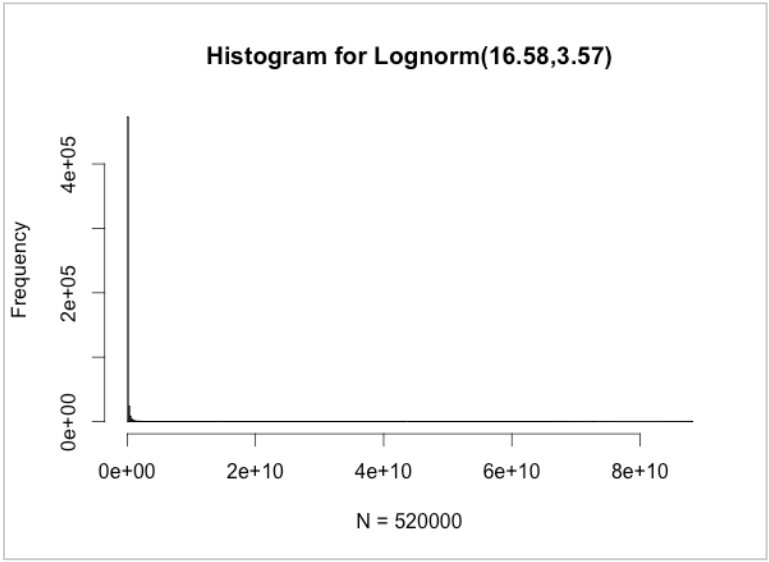

Figure 4-31 Histogram of individual log-normal costs for a sample of 1 million.

The next issue is how well the Bernoulli arrival process approximates the Poisson, both in terms of the number of arrivals and in terms of the distribution of total cost per decision epoch, which dictates the likelihood that the total cost might exceed the available budget.

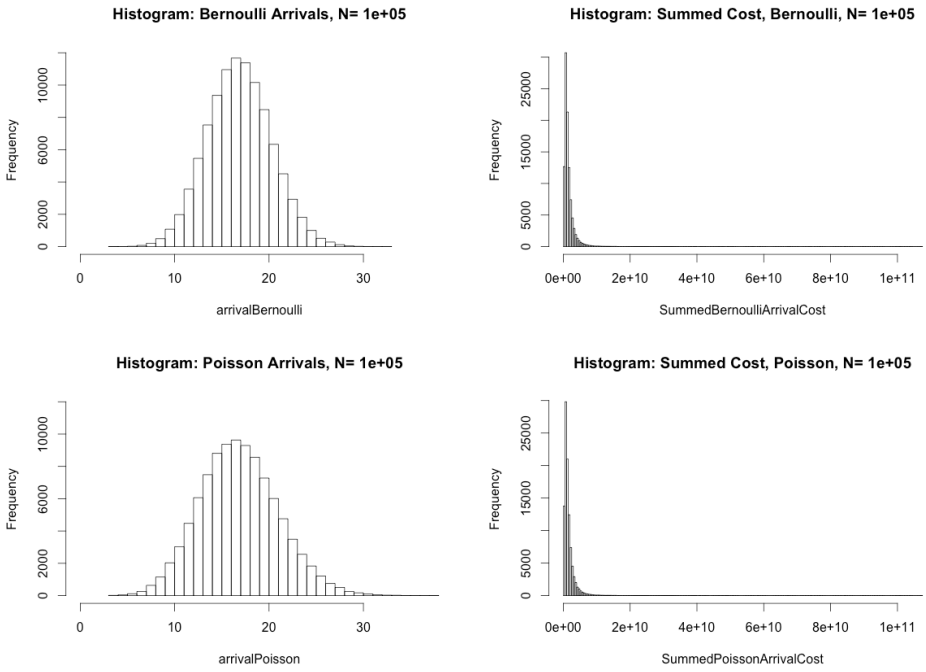

Figure 4-32 Panel Comparing Poisson Arrivals to Bernoulli Arrivals and their Respective Summed Cost Histograms

The extreme tails make comparison difficult. We are able to estimate numerically the likelihood that the summed cost will exceed the budget of 2.5B in the base case. For the Poisson arrivals the probability 0.159 and for the Bernoulli it is 0.157. We note two

things: there is a much higher probability in this case than in sample case that in a given epoch the cost of arrivals will exceed the available budget; and the Bernoulli arrivals result is virtually indistinguishable from the Poisson. This latter result is attributable to the four-fold increase in the number of time steps – which theory tells us results in increased convergence by the binomial approximation towards the Poisson.

Run Time. In the standard DP implementation, the cardinality of a state variable has profound effect on run time because the algorithm must visit every state. Without a means of approximating the value function, our initial ADP approach was worse, since we had to visit every state multiple times in order to calculate a means. Thus, the main technical obstacle for addressing the large-scale problem was the issue of run time.

The recursive regression value function approximation approach we have good reason to believe will change this. Theoretically, the scaling of the magnitude of the budget – whether 100 dollars or $2.5B – should have no effect on run times in our recursive regression ADP implementation. Using a budget of multi-billion resource units – where the fundamental resource unit that differentiates between buy and no buy could be a single dollar – is clearly a case of spurious precision. However, just as proof of principle we will use dollars to test the effect on run time. If correct, we should expect the only change in run time to be an increase from the number of stages, from twelve stages in the sample problem of 12 time stages to 52 stages representing the weeks in a year. The following table compares run times for these different approaches.

Table 4-8 Comparing Run Times of Different Approaches and Effect of Larger Size Problems on Recursive Regression Value Function Approximation

Approach	Stages	Budget	Runs	Time	Comments
DSKP	12	100	na	73 sec	Upper-bound of arcs: 20*100*100*12 = 2,400,000
Look Up Table Value Function	12	100	12,000	245 sec	Outer loop procedure: 7 outer loops
Recursive Regression Approx.	12	100	12,000	13 sec	
Recursive Regression Approx.	52	100	12,000	54 sec	Shows the effect of increased stages: essentially linear
Recursive Regression Approx.	52	2.50E+09	12,000	59 sec	Effect of large numbers on computational arithmetic?

Using representative JIEDDO data from the time frame we first examined this problem – which provides a kind of worst case scenario – we showed that the ADP with value function approximation solved this very large problem with relative ease.

Next we examine the quality of the resulting solution.

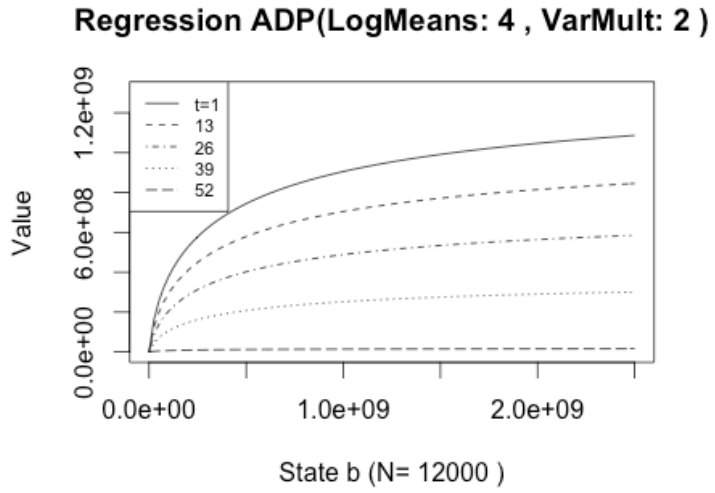

Figure 4-33 Cumulative Value Curve Using Large Scale Problem Data – Preliminary Solution

This curve shows that the problem data falls into the category of being mildly constrained. As one of their senior acquisition officials relayed to us at that time, "We are not really resource constrained." We must ask whether the apparent lack of concavity is normal. The lack of a point of comparison for this large-scale solution makes the ADP solution suspect.

4.4.4 Effect of Varying Variance

At first blush, it is natural to be concerned with the scale of the problem: costs for these decisions range in magnitude from hundreds of thousands to billions of dollars. However, when we make the units millions of dollars, the scale really becomes one of three orders of magnitude – from one million to one billion, or equivalently one to one-thousand. The example problem was concerned with two orders of magnitude. We could just as easily have made the base unit tens of millions of dollars and mimicked the scale of the example problem. But the other difference in these data is the variance of these two distributions. The example problem had a variance in log space of 0.5. The example problem has variance in log space of 3.5.

Lemma: Rescaling the distribution in linear space changes the mean but not the variance.

Proof. Let $X \sim \text{LogNormal}(\mu, \sigma^2)$.

If $Z = \log(X)$ then $\text{Exp}[Z] = \mu$ and $\text{Var}[Z] = \sigma^2$.

Let $Y = X/k$ – the rescaled version of X.

Then $\log(Y) = \log(X/k) = \log(X) - \log(k)$,

Which implies that $\text{Exp}[\log(Y)] = \text{Exp}[\log(X) - \log(k)] = \mu - \log(k)$.

But $\text{Var}[\log(Y)] = \text{Exp}[\log(Y)^2] - \text{Exp}[\log(Y)]^2$

➔ $\text{Exp}[\log(X)^2 - 2\log(X)\log(k) + \log(k)^2] - (\mu^2 - 2\mu\log(k) + \log(k)^2)$

➔ $\text{Exp}[\log(X)^2] - 2\mu\log(k) + \log(k)^2 - \mu^2 + 2\mu\log(k) - \log(k)^2$

➔ $\text{Exp}[\log(X)^2] - \mu^2 = \text{Exp}[Z^2] - \text{Exp}[Z]^2 = \text{Var}[Z]$.

Thus, rescaling a log-normal does not change its variance. Our initial focus on the need to scale appropriately was misplaced. The scale of the sample problem was sufficient – what was necessary for this particular distribution was to appropriately reflect the variance. The long right tail of the log-normal, particularly in cases of high variance, may be creating special difficulties for calculating the expectation of a dynamic stochastic knapsack. In the next section we revisit the sample problem but explore the effect of differing levels of variance.

The Effect of Differing Levels of Variance on Solution Methods

The main goal of this section is to revisit our comparison of DSKP and ADP solution methods. The sample problem had variance of 0.5 in log-space while the "full" problem data had log-space variance of 3.5. Our run matrix will include these two end-points plus solutions at 1.5 and 2.5 times the log-variance. In conducting the trial runs to prepare for this comparison, we discovered two things.

First, the increase in variance greatly slowed down the DSKP algorithm. This was not wholly unexpected but the increase in run time was startling and highlighted Dynamic Programming's Curse of Dimensionality.

The second discovery was that our sequence of log-means for the basis functions was ill adapted to the increase in variance. Recall, that our sequence means started on the mean of a single arrival and had as its end point the product of this mean and the expected number of arrivals. The steps between the log-means were the ratio of this distance in log-space to the number of required terms. However, as variance increased this sequence did not adequately address the resulting value functions and the fit suffered compared to the DSKP solution.

Noting that the mean of the log-normal is $\text{Exp}(\mu + \sigma^2/2)$, we experimented with using sequences where the steps in the sequence were multiples of log-variance $\sigma^2/2$ and the square root of this term. We finally settled on using the latter term as producing a fit that adapted to both low and high variance cases.

The table below compares the two approaches using two different cases of log-variance in the data. The case assumes a log-mean of 2, 4 expected arrivals and 4 terms in the log sequence. The variance-based sequence generates a similar sequence to the means-based sequence when variance is small but when variance it large, it expands the sequence accordingly. Note that the sequence starts not with the $\mu + \sigma^2/2$ but with $\mu + \sigma^2/2 + \sigma^2$; this means that first term in the regression would result in a mean larger than that of the single arrival.

Table 4-9 Comparing two methods for generating basis functions means, the old method depending upon the mean and number of the arrivals, the new strictly using variance/2

Variance	Method	Step	Log-mean 1	Log-mean 2	Log-mean 3	Log-mean 4
0.5	Old	0.2773	2.5273	2.8045	3.0818	3.3590
0.5	New	0.5000	2.7500	3.2500	3.7500	4.2500
3.5	Old	0.2773	4.0273	4.3045	4.5818	4.8590
3.5	New	3.5000	7.2500	10.7500	14.2500	17.7500

Note: Employes log-mean of 2, 4 expected arrivals, and 4 terms in the log-mean sequence

While this might seem counter-intuitive, this was an adjustment we made based on the observation that the recursive regression seemed to place a lot of emphasis on the first term. Forcing the first term to be small, especially as the variance got larger, resulted in a poor fit. The table below compares the resulting log-mean sequences

Armed with this new approach we set about completing the comparison of the ADP and DSKP approaches for the aforementioned four cases. We compare them graphically, via run time, and the value at the boundary conditions (t=1, b=B). This last is important because the solution at the boundary conditions is what informs the current decision.

Case 1: Variance = 0.5

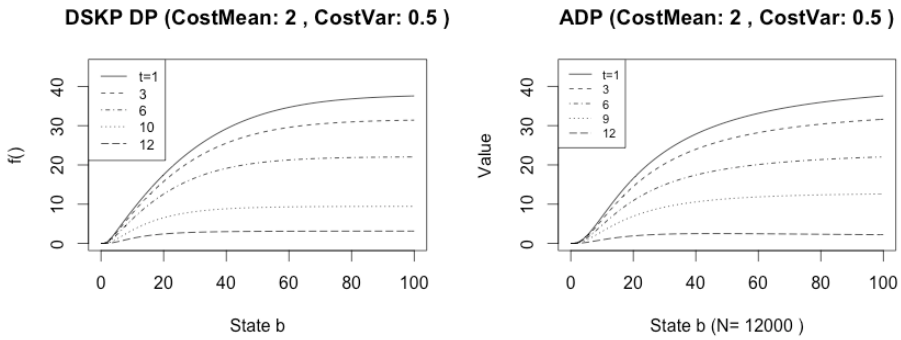

Figure 4-34 Comparing DP and ADP for Cost Log-Variance of 0.5

The solution times for the DSKP took 70 seconds while the ADP solution required 11 seconds. Qualitatively we see that the curve shapes are not exactly the same. Both share the early period of concavity but it is more pronounced in the ADP. The boundary condition value (t=1, b=100) for the DP approach was 37.54 while the ADP provided a starting value of 38.43.

Case 2: Variance = 1.5

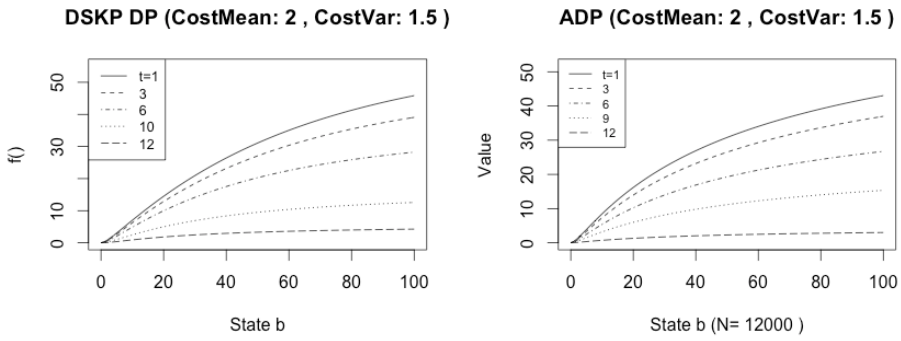

Figure 4-35 Comparing DP and ADP for Cost Log-Variance of 1.5

For the DSKP solution the additive increase in log-variance resulted in a 10 fold increase in run-time: the solution required 718 seconds. The ADP solution required increased to 16 seconds. Qualitatively, the ADP solutions tend to show more convexity, while the period of concavity has diminished for both. Boundary condition values were for DP 45.65 and for ADP 45.63.

Case 3: Variance = 2.5

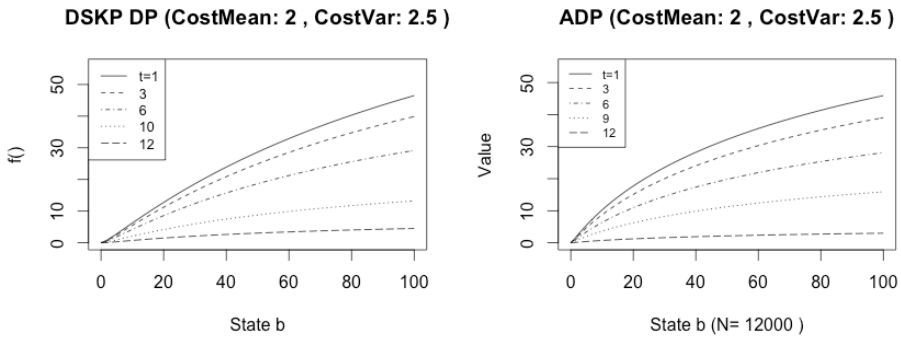

Figure 4-36 Comparing DP and ADP for Cost Log-Variance of 2.5

Once again the ADP solution was basically unchanged with a solution time of 11 seconds. For the DSKP the same increase in log-variance resulted in another order of magnitude increase in run-time. The solution required 8974 seconds. While we see more convexity from the ADP solution there is very little concavity left in either graph. The value of the boundary condition for the DP was 46.20 and for the ADP was 44.54.

Case 4: Variance 3.5

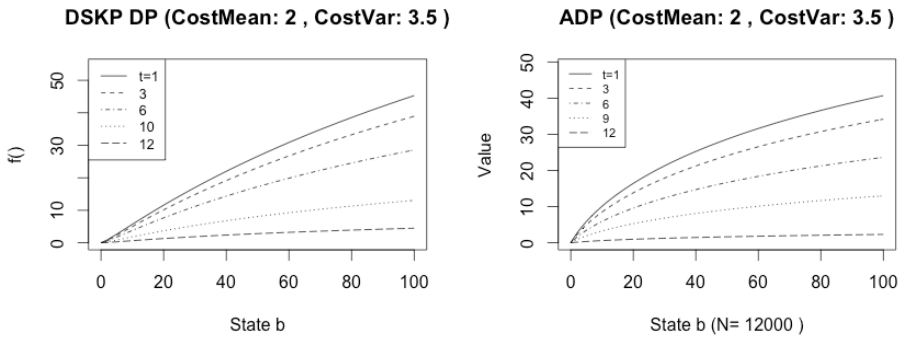

Figure 4-37 Comparing DP and ADP for Cost Log-Variance of 3.5

DSKP run time exploded to 126,361 seconds. ADP run time was 11 seconds. The ADP solution retains more pronounced convexity versus the DSKP solution. DSKP DP starting value was 45.29, while the ADP solution was 40.45.

Intuition tells us that the increase in variance should have led to a higher value for this case in the DP solution over the lower variances cases. However, value has dropped off slightly. It is our suspicion that the errors are numerical and may be a failing of the discretization scheme to effectively capture the effect of the very long tail of the log-normal with this high a variance. This same issue appears to have affected the ADP solution as well. This emphasizes the difficulty of effectively hedging against the effect of rare events.

Obtaining a good fit through the whole state-space for a log-normal cumulative curve of unknown parameters via recursive regression is obviously a challenge. We make no argument that the approach here is the best. However, at the starting conditions it provided a useful solution.

4.5 Summary

In this chapter we provided a brief introduction to DP and considered how to use DP to solve the binary knapsack problem. We introduced stochasticity and considered different variations on the static stochastic binary knapsack in order to build concepts we would need for the dynamic stochastic knapsack problem (DSKP), the simplified general case of the WPSP. We then delved into the DP approach for the DSKP developed by Papastavrou, Rajagopalan, and Kleywegt. While the WPSP's log-normal distribution of cost violated the consistency requirement for the DSKP, the violation was small and did not affect the quality of the solutions. We learned, however, that relying on the recursion to solve the WPSP lead to the "curse of dimensionality". The tremendous variance in the observed log-normal distribution would require hours if not days of computation time to solve useful instances of the problem.

ADP offered a solution approach. Instead of stepping backwards in order to calculate the value function by visit every node in a potentially huge discrete network representation of the entire state space, ADP generates samples of likely paths through the networks and learns the value function.

Implementing this vision required a lot of computational work, the greatest challeng being the implementation of machine learning. Learning requires a way of statistically estimating the value function. Because each problem is different, the method that works best requires much customization and adaption. In the WPSP, the log-normal distribution of cost of value led to the realization that the shape of value function as a function of state would be a cumulative log-normal, only we did not know the parameters. Thus, the machine learning became the challenge of recursively fitting a regression model that consisted of the linear combination of various log-normal basis functions. This effort yielded value functions for the full-scale problem that required about ten seconds of computation time (on average), in contrast with hundreds of thousands of seconds for the DP approach.

4.5.1 Implementing an ADP Solution

For those with an affinity for scientific computing, the lessons learned here from developing and coding an ADP approach for solving specific instances of the DSKP may be useful. Based on what we have learned the basic steps to implementing this approach for a similar dynamic stochastic knapsack problem are:

1. Implement a small scale solution via recursive DP. This is important to develop intuition to the nature of the problem and for validating your initial ADP solution.

2. Determine the shape of the value function. For the DSKP, one works to find an appropriate cumulative distribution. Learning will entail recursively fitting a set of basis functions.

3. Develop the ADP solution for the small scale problem. For the DSKP, the double pass algorithm was best.

4. Begin with a single basis function while you develop the best step-size approach for your problem.

5. Validate your ADP results with the small scale DP solution prior to implementing the approach on full-scale problems.

4.5.2 The Nature of the ADP Solution

Overall the quality of solutions produced by ADP still leaves something to be desired compared to the exact DP approach. In fairness, ADP is just that: an approximation. It is important also to recognize that the big differences in results between the two approaches arose in the portion of the state-space farthest from the starting conditions: time "now" and the current budget. This is not a large impediment since we are interested in the decision to be made *today* but need to account for the impact of likely decisions tomorrow. In support of a decision, we would like to rerun the model with current data versus relying on a look-up table that was created several time steps before and at a different resource level.

This way of looking at the problem – decisions today (boundary conditions) versus decisions that must be made in the future - provides a segue to our next topic: stochastic programming.

CHAPTER 5 – SOLUTION METHODS: STOCHASTIC PROGRAMMING

5.1 Stochastic Programming Introduction – The Two-Stage Problem

Stochastic programming has its roots in linear programming. As described previously, Dantzig (1955) and Beale (1955) independently described linear programs with uncertain elements. In his paper, Dantzig described the essence of what is now the standard two-stage stochastic problem.

We draw, with modifications to suit our purposes, on Birge and Louveaux for their formulation of the two-stage program. The central modification is our perspective of value maximization subject to resource constraints as opposed to minimizing costs subject to meeting requirements. The two-stage formulation provides the fundamental insight into a variety of planning problems. There is a given set of the facts we know today: current constraints on choices A, values $\boldsymbol{v}$, and current resource bounds $\boldsymbol{b}$. The choices we make today we describe via the decision variable $\boldsymbol{x}$.

There are beliefs we hold about the future. We envision our decisions occurring over an epoch, beyond which we will not extend our analysis. Our beliefs about the future we describe via a random vector $\boldsymbol{\xi}$, whose realizations we indicate by ω. This future encompasses constraints $FA(\omega)$, values $\boldsymbol{fv}(\omega)$, resource bounds $\boldsymbol{fb}(\omega)$, and, linking our present decisions to the future, the matrix $T(\omega)$. Compactly, $\boldsymbol{\xi}(\omega) = (FA(\omega), T(\omega), \boldsymbol{fv}(\omega), \boldsymbol{fb}(\omega))$. For decisions in the future we use the variable $\boldsymbol{y}$. We wish to maximize

total value over the decision epoch, the benefits to be gained today and those in the future. Since the future is uncertain, we are maximizing an expectation over $\boldsymbol{\xi}$.

General Two-Stage Stochastic Program (G2SSP):

Maximize $\quad \boldsymbol{v^T x} + \mathbb{E}_{\xi}[\boldsymbol{f v^T y}]$

Subject to $\quad A\boldsymbol{x} \leq \boldsymbol{b}$

$$FA(\omega)\boldsymbol{y} + T(\omega)\boldsymbol{x} \leq \boldsymbol{f b}(\omega)$$

$$\boldsymbol{x}, \boldsymbol{y} \geq \boldsymbol{0}$$

The objective function contains both deterministic and stochastic elements, the latter addressed via an expectation. The constraint structure is two fold, representing deterministic and random constraints. The deterministic constraints are presented in standard canonical form for maximization.

The random constraints contain two matrices. Birge and Louveaux label *FA* as the *recourse matrix* and *T* as the *technology matrix.* The name for the former is self-evident. In the future, the random vector $\boldsymbol{\xi}$ has been realized and *FA* defines our recourse: it provides the coefficients for the constraints on our recourse decisions. The origin of the term "technology matrix" is murky. Our interpretation is that "technology" conveys the extent of our knowledge on the eventual impact of present decisions. The better our technology, the more we understand what impact our decisions will have. In the two-stage context, "perfect" technology would arguably lead to a deterministic *T*

matrix. Another way to look at the two-stage problem is via the so-called deterministic equivalent.

Deterministic Equivalent of the General Two-Stage Stochastic Program

Maximize $\boldsymbol{v^T x} + Q(\boldsymbol{x})$

Subject to $A\boldsymbol{x} \leq \boldsymbol{b}$

$\boldsymbol{x} \geq \boldsymbol{0}$

Where

$$Q(\boldsymbol{x}) = \mathbb{E}_{\xi}[Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega))]$$

and

$$Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega)) = \max_{\boldsymbol{y}}\{\boldsymbol{f v^T y} | FA(\omega)\boldsymbol{y} \leq \boldsymbol{fb}(\omega) - T(\omega)\boldsymbol{x}, \boldsymbol{y} \geq \boldsymbol{0}\}.$$

Birge and Louveaux call $Q(\boldsymbol{x})$ the *value function* of $\boldsymbol{x}$. This formulation highlights that any decision $\boldsymbol{x}$ seeks to maximize a sum of current value, which faces constraints and future value which depends on $\boldsymbol{x}$. The general idea is that the random vector $\boldsymbol{\xi}$ has a finite index set of realizations Ω, where each realization $\boldsymbol{\xi}(\omega)$ is indexed by ω ε Ω, each with probability p_ω. This means that the value function can be expressed as a weighted sum of the solutions of these realizations, or scenarios.

$$Q(\boldsymbol{x}) = \sum_{\omega=1}^{|\Omega|} p_\omega Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega)) \text{, where } \sum_{\omega=1}^{|\Omega|} p_\omega = 1 \text{ and } p_\omega > 0 \ \forall \omega.$$

5.1.1 Feasibility Sets

In two-stage problems there needs to be a particular concern with whether decisions in the first stage may result in an infeasible second stage. Our discussion again relies on Birge and Louveaux. The value function $Q(\boldsymbol{x})$ clearly depends on $\boldsymbol{x}$. As shown above, $Q(\boldsymbol{x})$ is the weighted average of individual scenario solutions $Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega))$. If some $\boldsymbol{x}$ for some scenario $\boldsymbol{\xi}(\omega)$ results in $Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega))$ being infeasible, by convention we let $Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega)) = -\infty$. To make our convention complete (since we are maximizing), we have that $+\infty + -\infty = -\infty$. That is, when we calculate the value function of $\boldsymbol{x}$, we do not want the possibility of one infinitely good scenario to somehow outweigh the presence of an infeasible one. Thus, any infeasible scenario $Q(\boldsymbol{x}, \boldsymbol{\xi}(\omega))$ makes $Q(\boldsymbol{x})$ infeasible. With these conventions in place we define two feasible sets, one for the first phase and one for the second.

Definition:

$$K_1 = \{ \boldsymbol{x} \mid A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0} \}$$

$$K_2 = \{ \boldsymbol{x} \mid Q(\boldsymbol{x}) > -\infty \}$$

This allows us to rewrite the deterministic equivalent as follows:

Maximize $\quad \boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{x} + Q(\boldsymbol{x})$

Subject to $\quad \boldsymbol{x} \in K_1 \cap K_2$

Note that both feasibility sets depend on $\boldsymbol{x}$. Implicit is the idea that we may find an $\boldsymbol{x}$ in K_1 that is not in K_2.

5.1.2 Special Cases

In the general case we described above, we have that all second stage data are stochastic. It is useful to identify special cases, which are identified by what portions of the second stage are random and their theoretical effects on feasibility. Birge and Louveaux provide a thorough discussion on the conditions associated with different cases of the two-stage problem and their affect on determining theoretical feasibility. We provide a highlight of these, while marching towards our goal of being able to use stochastic programming to address the WPSP.

5.1.3 Fixed Recourse and Random Technology

The standard two-stage problem, as presented originally by Beale and Dantzig and employed throughout most of Birge and Louveaux, assumes fixed recourse; that is, that FA is deterministic while the technology matrix contains the random elements. When the recourse matrix is fixed and the random vector is finite or has finite moments, the stochastic program is convex and in general amenable to a variety of solution methods. Within fixed recourse there are some additional conditions worth noting.

Relatively Complete Recourse. This is the case when every $\boldsymbol{x}$ that satisfies the first phase constraints also satisfies the second phase constraints. That is, $K_1 \subset K_2$. Unless you have a case of special structure, it may be hard to determine in general if relatively complete recourse exists.

Complete Recourse. This is a special case of relatively complete recourse where we can test whether $K_1 \subset K_2$. Complete recourse holds when there exists some $\boldsymbol{y} \geq 0$ so that $FA\boldsymbol{y} = \boldsymbol{t}$ for all $\boldsymbol{t}$ in $\boldsymbol{R}$.

5.2 Basic Approach

When we first encountered what we would later describe as the WPSP, we were struck by the following insights:

- The urgency of wartime requires that solutions be acquired as soon as possible.
- Bad choices today reduce the ability to acquire better solutions tomorrow.

This simplistic but powerful partitioning of the sequence of decisions between the deterministic "now" and the stochastic "tomorrow" led to the idea to formulate this as a two-stage stochastic integer program.

We lay out our assumptions about the future, which are the same as employed in the WPSP formulation in the previous chapter on the Dynamic Programming approach.

- Initiatives arrive via a constant-rate Bernoulli arrival process; that is, in every time unit a single unit arrives with probability p.
- After an initiative arrives, its attributes and costs become known.

- We assume the existence of an accepted method to translate the attributes of any initiative into a common measure of military value, and assume that these military values are additive.
- Based on past history, we can determine a joint distribution of future initiative costs and values.
- For a given finite period of time into the future, we represent this joint probability distribution as the random vector ξ. An instance of this random vector is a possible future. For clarity, a possible future will consist of three ordered sets: an index set of future time steps where in each time period a initiative can arrive with probability p $\{k \mid k=1,\ldots,K\}$, the set of their costs $\{futCost_k\}$, and the set of their values $\{ futValue_k \}$, where in any time step k where there is no arrival
- Future initiative costs are log-normally distribution.
- Future initiative value depends upon the cost of the initiative. Given the cost $futCost_k$ of the k-th arrival, its value $futValue_k$ is drawn from a uniform distribution $U(0, 2\ futCost_k)$, with mean equal to the sampled cost of the arrival.

Note that the decision horizon is finite – a natural assumption given that governments base many of their discretionary fiscal decisions on the available funds from an annual budget. Also, while the original process we observed behaved more like a

Poisson process, we use a Bernoulli process to approximate a Poisson arrival process and thereby bound the total possible number of arrivals in the future.

In most weeks, the decision to fund the recently arrived initiatives is feasible: The budget far exceeds the cost of any one initiative. Intuitively, the decision should consider both the present demands for funding (the value and cost of the recently arrived initiatives) and the future demands that may yet arrive before the end of the funding period.

To illustrate our approach to modeling this arrival process we provide an example of a set of samples drawn from the random vector.

Table 5-1 Scenario Data Example

Costs

Scenario\ Wks	1	2	3	45	46	47	48	49	50	51	52
1	0	0	1	4	6	0	0	0	161	0	38
2	8	0	0	0	310	0	1	0	0	0	0
3	0	0	0	4	0	0	0	0	18	293	0
4	0	50	4	0	0	10	13204	0	0	0	289
5	0	0	0	0	0	0	0	30	0	9	0

Values

Scenario\ Wks	1	2	3	45	46	47	48	49	50	51	52
1	0	0	0	1	1	0	0	0	289	0	58
2	3	0	0	0	425	0	3	0	0	0	0
3	0	0	0	5	0	0	0	0	9	180	0
4	0	58	8	0	0	2	15018	0	0	0	230
5	0	0	0	0	0	0	0	52	0	7	0

The table consists of a 52 week year where we have drawn 5 instances from the random vector. For brevity we obscure all but the beginning and ending weeks. Cells with a zero show weeks where there was no arrival. The costs are log-normally distributed: note the large range in values. The arrivals' values are drawn from a uniform distribution that ranges from 0 to twice the corresponding cost, resulting in a distribution mean equal to the cost. The resulting values are values that are on average proportional to their costs. As described in Chapter 4, this results in random values that are on average proportional to cost.

These assumptions about the future are clearly very simplistic, but they are based in part on the observations we summarized in Chapter 1 and are consistent with the assumptions we used for the approaches outlined in Chapter 4. The stochastic programming approach certainly allows for more nuanced representations of the future, such as removing assumptions about independence between arrivals, since all it needs is a process for generating a representative set of scenarios.

5.2.1 Formulation

We start with a formulation WSPS as a two-stage integer stochastic program. The reader may ask why not employ an n-stage formulation of the stochastic binary knapsack. But in essence we have already dealt with this as the dynamic programming representation of the WPSP. Here we employ the two-stage formulation as a useful and easy to solve approximation. We justify our approach by noting that we can solve

determinist binary knapsacks as either a dynamic program – a very simple n-stage problem – or as an single stage integer program. This sample logic can be carried over to stochastic knapsacks, where we will collapse the n-stages into a initial deterministic stage (the "here and now") and a second stage ("the future"), that consists of the remaining n-1 stochastic stages.

The first stage represents the present decision; that is, in the current week of the funding period, choose from among the set of recent arrived initiatives which to fund. The second stage – the recourse stage - consists of the future initiatives from which to choose in the remainder of the funding period with a budget decremented by the first stage decision. The future is random so we will represent it by generating a set of possible futures; the sample is used to approximate the distribution of the random vector ξ. The objective is to maximize expected value across both stages.

Two-Stage Stochastic Integer Program formulation of the WPSP (WPSP2SSIP)

Indices

i	recently arrived initiative in set of current initiatives I
s	possible future scenario in set of scenarios S – the index of the sample set
k	time steps remaining in the decision epoch

Data

$Cost_i$	cost of recently arrived initiative i

$Value_i$	value of recently arrived initiative *i*
$futCost_{sk}$	cost of future initiative *k* in scenario *s*
$futValue_{sk}$	value of future initiative *k* in scenario *s*
P_s	probability of scenario *s*
Budget	available budget

Variables

Buy_i	binary decision variable, 1 if initiative *i* is funded, 0 otherwise
$futBuy_{sk}$	binary decision variable; 1 if initiative *k* in scenario *s* is funded, 0 otherwise

Objective

Maximize

$$\sum_i Value_i * Buy_i + \sum_s P_s \sum_k futValue_{sk} * futBuy_{sk}$$

Constraints

Stay within Budget

$$\sum_i Value_i * Buy_i + \sum_k futCost_{sk} * futBuy_{sk} \leq Budget, \forall s$$

The budgetary constraint is intuitive – under any given future we must stay within budget. Additional constraints that could enrich model include constraints that address interactions amongst first stage initiatives and constraints aimed at satisfying strategic goals within the portfolio.

How does this formulation map back to the G2SSP? Recall the general formulation of the two-stage problem.

G2SSP

Maximize $\boldsymbol{v^T x} + \mathbb{E}_{\xi}[\boldsymbol{f v^T y}]$

Subject to $A\boldsymbol{x} \leq \boldsymbol{b}$

$FA(\omega)\boldsymbol{y} + \boldsymbol{T}(\omega)\boldsymbol{x} \leq \boldsymbol{fb}(\omega)$

$\boldsymbol{x}, \boldsymbol{y} \geq \boldsymbol{0}$

The $\boldsymbol{v}$ and $\boldsymbol{fv}$ vectors are represented by $Value_i$ and $futValue_i$. The recourse matrix $FA(\omega)$ is matrix of $futCost_{sk}$, which stands in contrast to the deterministic recourse matrix in the standard two-stage fixed recourse problem. The budget is fixed throughout so $\boldsymbol{b} = \boldsymbol{fb}$. The technology matrix – the effect of $\boldsymbol{x}$ on future scenarios – is the deterministic $Cost_i$. In the deterministic equivalent, and when we decompose the problem, it is useful to think about the matrix A as also consisting of $Cost_i$ so that $A = T$. This makes $Ax \leq \boldsymbol{b}$ redundant. Thus, our basic formulation does not include this first stage constraint. If we extended the problem to consider different first stage constraints – for example, interaction and satisficing constraints – then by necessity these would be included in the first stage constraint $Ax \leq \boldsymbol{b}$.

In the classic two-stage thinking, though, this makes no sense. We may have a particularly expensive initiative recently arrived in the first phase – one that exceeds the current budget – so that if we did not have the budget constraint in the first stage we

might make an acquisition that makes the second phase infeasible. While this does not change the fact that the constraint is redundant and any solver would promptly drop it, we need this constraint when we formulate the deterministic equivalent of the WPSP.

Deterministic Equivalent of the WPSP2SSIP

Maximize $\boldsymbol{Value^T Buy} + Q(\boldsymbol{Buy})$

Subject to $\boldsymbol{Cost^T Buy} \leq Budget$

Where

$Q(\boldsymbol{Buy}) = \mathbb{E}_S[Q(\boldsymbol{Buy}, s)]$

and

$Q(\boldsymbol{Buy}, s) = \max_{\boldsymbol{futBuy_s}}\{\boldsymbol{futValue_s}^T \boldsymbol{futBuy_s} | \boldsymbol{futCost_s}^T \boldsymbol{futBuy_s} \leq Budget - \boldsymbol{Cost^T Buy} \geq \mathbf{0}\}$.

The utility of the deterministic equivalent approach is that it provides the key insight that our goal is to optimize total value – the value of the decision today and the future value this decision is expected to yield. As described earlier, $Q(\boldsymbol{x})$ is called the value function of x.

As discussed earlier, two-stage problems without fixed recourse create special problems in establishing whether complete recourse exists. It should be relatively apparent that in the case of the WPSP SIP formation, since the first phase will never be allowed to exceed budget, then the second phase will always be feasible – particularly since buying nothing in the second phase is still a feasible answer. One question then is how to generate these future scenarios.

5.2.2 Monte Carlo Scenario Generation and Sample Average Approximation

As described in the assumptions above, a scenario consists of a set of arrivals of random cardinality, with for each arrival a random pair of cost and value, which come from a joint-distribution. This is not a finite distribution, since the values of costs and values have continuous distributions. Our intended approach is to use Monte Carlo simulation to generate a set of futures and solve the resulting stochastic integer program (SIP). While this approach seems natural enough, questions arise. Is there a theoretical foundation for this approach? With larger sample sizes we expect better solutions, but given the nature of integer programming we also expect exponentially increasing difficulty in solving them. Is there an ideal sample size? Since we are solving an expectation over a sample, is there a way to characterize the quality of our solutions?

Kleywegt et al (2001) examine the use of Monte Carlo simulation as an approach to solve stochastic programs of the following general form.

General Discrete Stochastic Program (GDSP)

$\min_{x \in S} \{ g(x) \coloneqq \mathbb{E}_{\xi} G(x, \xi) \}$

where

$\mathbb{E}_{\xi} G(x, \xi) = \int G(x, \xi) d\xi$ and S is finite.

They assume that the random vector has a continuous distribution with finite moments. Their intent is to develop an algorithm for solving these problems using Monte Carlo simulation to generate N i.i.d. samples from the random vector: $\xi^1,\ldots,\xi^N$. This leads to the sample average approximation to the stochastic problem.

Sample Average Approximation (SAA) to GDSP

$$\min_{x\in S}\left\{\hat{g}(x) \coloneqq \frac{1}{N}\sum_{j=1}^{N} G(x,\xi^j)\right\}$$

They define u^* as the solution of the true problem and S^* the solution set of the true problem; and $\hat{u}_N$ as the solution to the SAA and $\hat{S}_N$ as the corresponding solution set. Note that $\mathbf{E}(\hat{u}_N) = u^*$. They build theory supporting a Monte Carlo strategy, of which we provide a brief overview.

They start by showing that for large N, $\hat{u}_N$ converges exponentially on u^* but that the rate depends upon many things including sample size and κ – a condition number they define for discrete stochastic problems that provides a rough measure of the likelihood that the sample N will result in $\hat{u}_N = u^*$. In proving the exponential convergence they rely on the use of the tolerance parameter ε. They define the concept of the ε-optimal solution set $\hat{S}_N^{\varepsilon}$: all solutions $\hat{u}_N$ where $|\hat{u}_N - u^*| < \varepsilon$. They show that the error term $\hat{u}_N - u^*$ is normally distributed with mean zero and with variance equal to the variance of the objective function. Prior to laying out their algorithm, they discuss

practical considerations such as the opposition between, on the one hand increasing N to improving solution quality, and on the other hand decreasing N to improve the solution time.

We use their results to create a simplified version of their algorithm built on experience with the WPSP SIP and our experience with the stochastic gradient algorithm from the previous chapter. Recall the recursive mean updating equation from Chapter 4 where θ^n is the mean at the n-th iteration and $R(\omega)$ is the latest sample.

Equation 5-1: Mean Updating Equation

$$\theta^n = \frac{n-1}{n}\theta^{n-1} + \frac{1}{n}R(\omega)$$

First we note that since $\mathbf{E}(\hat{u}_N) = u^*$, then for M iterations can define $\frac{1}{M} * \sum_{m=1}^{M} \hat{u}_N^m := \bar{\bar{u}}_N^M$ which as M increases converges on $\mathbf{E}(\mathbf{E}(\hat{u}_N)) = u^*$. Then define at the m-th iteration our estimate of the error, $e^m = |\bar{\bar{u}}_N^m - \bar{\bar{u}}_N^{m-1}|$. As M grows this is guaranteed to converge to zero, however, we also know that the convergence is not monotonic. So we should set ε to higher precision than needed in order to prevent premature convergence. Prudence dictates that we should start with a prefixed set of iterations M.

Simplified SAA Algorithm

1. Choose sample size N, initial set of iterations M, tolerance ε, and factor β.
2. For $m = 1,\ldots, M$:
    a. Generate sample of N instances of random vector.
    b. Solve SAA to find $\hat{u}_N^m$.
    c. Let $\bar{\bar{u}}_N^m = \frac{1}{m} * (\hat{u}_N^m + (m-1) * \bar{\bar{u}}_N^{m-1})$
    d. Let $e^m = |\bar{\bar{u}}_N^m - \bar{\bar{u}}_N^{m-1}|$
    e. Return $\bar{\bar{u}}_N^m$, if $e^m < \varepsilon$, else continue.
3. Let $M = \beta M$, return to step 2.

SIP Computational Details:

We employed the free-ware statistical language R to code the algorithms, to generate the samples from the random vector $\boldsymbol{\xi}$ and to formulate the SIP. Via Gurobi's interface with R, we called Gurobi from R. To handle the large size of some of the SIPs, we made extensive use of sparse matrices. The codes were all run on a MacBook Pro, with 2.3 GHz Intel Core i7 and 8 GB of 1600 MHz RAM.

Comparing the WPSP to the GDSP, we make a couple of observations. The WPSP is an example of a two-stage recourse problem. The first stage in the simplest form of this problem consists of a single decision – whether or not to acquire the latest arrival. This decision is weighed against the expected value of future arrivals. It is

determining this expected value that provides the necessary context for this binary decision: $Q(x=0) < Q(x=1)$?

The practical question to ask is to what precision is needed to answer this question? Since the decision is binary (we either buy or not buy), then for any initiative where the difference between $Q(x=0)$ and $Q(x=1)$ is infinitesimal, this simply reflects that we are indifferent as to whether we acquire or not. In the next section we explore the precision of the solutions obtained from the WPSP SIP implementation.

5.3 Numerical Solutions of the WPSP2SSIP

We explore numerical solutions generated by the WPSP2SSIP implementation starting with the example problem used in the previous chapter.

5.3.1 Example Problem Implementation

The sample problem used the following data.

Example Problem Data

T = 12 time periods

B = 100 resource units

p = 1/3 - likelihood of binary arrival in any given time step

$C \sim$ Log-normal(2, 0.5) distribution of cost given an arrival

$V \sim$ Uniform(0, 2C) distribution of value of an arrival given its cost

What we are trying to understand is the value function $Q(x)$. Thus for the first phase we use an arrival with value and cost both 0. Thus, solving the problem yields the first stage value of 0 plus $Q(x)$.

Effect of the choice of N on a single replication of the SIP

From past observation, we know that this system is only "loosely" constrained. In most samples of future scenarios, resources exceed constraints; that is, in most scenarios the solution is to buy all future arrivals. Below we show how the solution varies by the number of sampled scenarios for a single iteration of the SIP. Each sample is generated using the same random seed. The second column of table highlights how many scenarios require more total resources than available: we see that approximately 1.4% of the samples are constrained.

Table 5-2 Effect of Sample Sizes on WPSP SIP Solution for the Example Problem

N Scenarios	ScenarioCost>Budget	%Constrained	Solution	Time(sec)
100	0	0.0%	36.33	0.00
250	1	0.4%	36.83	0.01
500	4	0.8%	36.64	0.01
1,000	12	1.2%	37.09	0.02
2,500	40	1.6%	37.89	0.06
5,000	77	1.5%	37.58	0.11
10,000	149	1.5%	37.53	0.22
25,000	360	1.4%	37.60	0.71
50,000	695	1.4%	37.81	1.50
100,000	1351	1.4%	37.88	3.84

We employed a variance reduction technique by generating a very large single sample from the Monte Carlo simulation, i.e., we ran the simulation only once. From this data table, we selected from the start the required number of rows. Thus, when solving for N=100, 250,…, 50,000, all cases share the first 100 rows, all but the first share the first 250 shares, and so on. As N increased, the number of scenarios that had a total cost of arrivals exceeding the budget converged to approximately 1.4%, which is why we call this problem "loosely constrained".

Our best estimate for the true mean is 37.88 after 100K replications. We would like to know more about the variability of this solution. As a point of comparison, the value function for the Dynamic Programming implementation of the DSKP at t=1 and capacity 100 was 37.54, and the value function of the final ADP implementation at the same point was 38.43.

Solution times were very fast compared to our earlier efforts. Via R's integration with Gurobi and the extensive use of R's matrix operations and sparse matrices, not only did we obtain very fast solutions to these large, albeit loosely constrained, problems, but the entire run of 10 SIPs of increasing size took a total of 6 seconds. For the largest scenario samples, the 25,000 size sample instance yielded a solution in .7 seconds, the 50K in 1.5 second, and the 100K in 3.8 seconds. However, what appears to really be driving solution time is the number of constrained scenarios. We should expect that under less loosely constrained conditions we should see the solution times slow down, all else being equal.

To examine the effect of increasing variance, we examined a sample of the scenario sizes and compared these over different levels of log-variance. The results are presented on the table below.

Table 5-3 Results from increasing levels of Log-Variance and N for the Example Problem

Log-Var	N	Constrained	%Constrained	Solution	Time(sec)
0.5	250	1	0.4%	37.76	0.005
1.5	250	45	18.0%	47.30	0.009
2.5	250	70	28.0%	49.93	0.008
3.5	250	92	36.8%	49.78	0.012
0.5	2,500	29	1.2%	37.68	0.060
1.5	2,500	410	16.4%	47.81	0.075
2.5	2,500	753	30.1%	49.79	0.081
3.5	2,500	982	39.3%	49.39	0.075
0.5	25,000	351	1.4%	37.77	0.654
1.5	25,000	4250	17.0%	48.26	2.675
2.5	25,000	7704	30.8%	49.86	2.965
3.5	25,000	9947	39.8%	48.73	1.623

Increasing run times were seen from increasing log-variance within the number of scenarios, but the effect was weak compared to the effect of increasing the number of scenarios. Note that any measure we collect from solving an instance of a SIP is itself a random variable: the solution, the run time, the number of constrained solutions. Moving forward, we should use multiple replications to understand each of these random variables.

Effect of N on the Simplified SAA algorithm

Recall that in choosing ε we must select a precision higher than needed. The data in this current set has no meaningful unit but for sake of argument we say that our desired precision is a unit of resource – which equates to 1% of our budget. The tolerance for the algorithm should be of higher precision to avoid premature convergence, so choose an order of magnitude tighter ε: 0.1 resource units.

To explore the stochastic gradient, we decided to arbitrarily run the algorithm for a fixed number of iterations regardless of stopping conditions for a set where N the number of scenarios varied in size from 2,500 to 25,000. We then examined the iterations to see where we should have stopped for our desired precision. We show our results in both tabular form and graphical.

N	Rep	*u*	*u-hat*	*e*	*e* < 0.1
2,500	1	37.100	37.100		
	2	37.458	37.279	0.179	
	3	38.157	37.572	0.293	
	4	39.039	37.938	0.367	
	5	37.984	37.948	0.009	**stop**
	6	38.128	37.978	0.030	
	7	37.589	37.922	0.055	
	8	38.070	37.941	0.018	
	9	38.663	38.021	0.080	
	10	37.476	37.966	0.054	
10,000	1	37.932	37.932		
	2	37.945	37.938	0.006	**stop**
	3	37.647	37.841	0.097	
	4	37.949	37.868	0.027	
	5	37.886	37.872	0.004	
	6	37.794	37.859	0.013	
	7	38.116	37.895	0.037	
	8	37.567	37.854	0.041	
	9	37.670	37.834	0.020	
	10	37.744	37.825	0.009	
25,000	1	37.960	37.960		
	2	37.865	37.913	0.047	**stop**
	3	37.469	37.765	0.148	
	4	37.554	37.712	0.053	
	5	37.865	37.743	0.031	
	6	37.858	37.762	0.019	
	7	37.643	37.745	0.017	
	8	37.946	37.770	0.025	
	9	37.989	37.794	0.024	
	10	37.865	37.801	0.007	

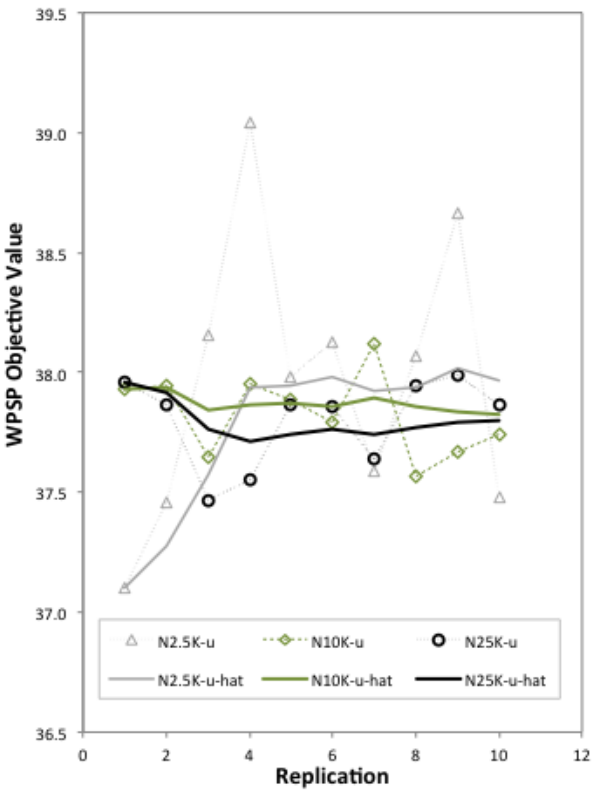

Figure 5-1 The effect of sample size on stopping conditions for the simplified SAA algorithm

The 2.5K sample required 5 iterations, that is, we generated and solved 5 instances of the problem before reaching the stopping criterion. Note that the error did not decrease monotonically, growing to exceed the stopping conditions a couple of times in this limited set of replications. The 10K and 25K samples both stopped on the second iteration. They both would have been roughly a tenth value unit from the *u-hat*[10] value, well within the desired precision.

As discussed in the previous section, any measure we obtain from running the SIP is itself a random variable. It then follows that the number of iterations required before the SAA meets the stopping conditions is itself a random variable dependent on the number of scenarios N.

To develop a stronger contrast among the choices for N, we decided to run the SAA ten times at a broader range of scenario samples, from 1,000 to 100,000. We tracked both the number of iterations required and the solution time for each run of the algorithm. From the ten runs of the algorithm, we calculated the average number of iterations required before the algorithm stopped and the average of the solutions obtained when the algorithm stopped.

Lastly, we also calculated a 95% confidence interval half-width around the mean of the 10 SAA runs to show the precision of the answer obtained by the algorithm at each case. Smaller half-widths indicate that the algorithm solution were closer together over the 10 runs. It could be that running the small problem a few iterations might find the answer to the same precision as the large problem in quicker time.

Table 5-4 Results from 10 runs of SAA at increasing levels of Log-Variance and N

LogVar	N	Avg Iterations	Avg Run Time	Solution	95%CI-HW
0.5	1,000	3.8	0.096	37.910	0.854
0.5	10,000	2.4	0.597	37.852	0.331
0.5	100,000	2.1	9.237	37.791	0.120
1.5	1,000	5.3	0.197	48.617	1.074
1.5	10,000	3.4	1.415	48.559	0.461
1.5	100,000	2.6	73.962	48.384	0.184
2.5	1,000	4.5	0.229	50.169	1.002
2.5	10,000	3.0	1.403	49.877	0.642
2.5	100,000	2.6	85.654	49.964	0.087
3.5	1,000	6.4	0.337	48.951	1.085
3.5	10,000	3.0	1.181	48.953	0.517
3.5	100,000	2.6	406.882	48.967	0.096

The SAA algorithm, for the SIP implementation of the WPSP, showed good solutions very quickly from small N solutions but the precision was not comparable to the larger N. There is a clear trade between precision and run time. In terms of trades between run-time and precision, the N=10,000 appears to be a good compromise.

The next issue to study is the shape of the value function as a function of remaining budget. We have seen that for the 0.5 log-variance cost, 100 unit-budget case, the DSKP, ADP, and SIP approaches all align with values in the range of 37.5-38.4.

$Q(x)$ as a function of B

The next issue to study is the shape of the value function as a function of remaining budget. We have seen that for the 100-unit budget case, the DSKP, ADP, and

SIP approaches all align with values in the range of 37.5-38.4. From the previous chapter we know that while the shape of the value function is largely concave, for very small budgets the curve becomes convex due to the pronounced left skew of the log-normal distribution. Essentially for these very small budgets, the shape of the value functions reflects that the cost of a single arrival, approaching the shape of the left tail of the log-normal distribution.

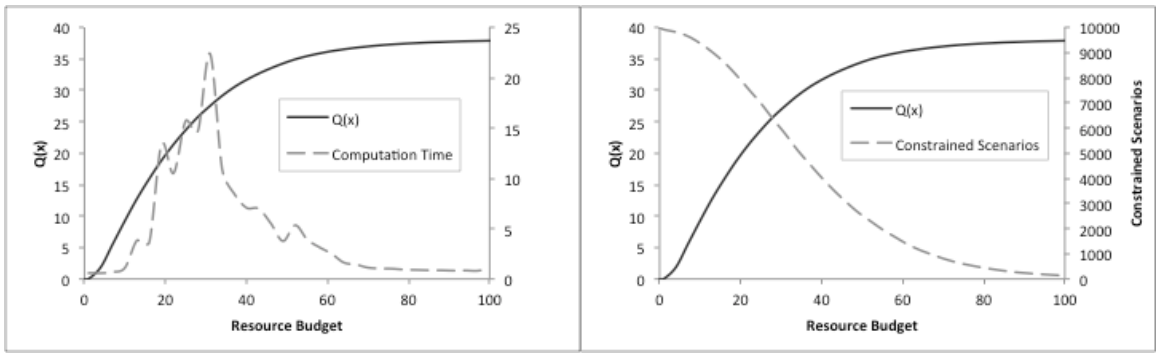

Figure 5-2 Q(x) vs Resource Budget, with computation time plotted on a secondary axis on the left chart, and the number of constrained futures plotted on the secondary axis on the right chart.

Before comparing these results to the dynamic programming results, we should note the solution times observed. Our intuition was that as the number of future scenarios that became constrained increase, solution times would slow down. This was initially true. When the number of constrained scenarios increased to about 50%, the solution times were highest. However, as the constrained cases increased further, the solution

times reduced from this peak time. Finally, when the system was severely constrained, solution times were the same as when the system was essentially unconstrained.

The very swift solution times when the system is largely unconstrained and when the system is severely constrained are the result of pre-processing. Pre-processing searches for special cases such as redundant constraints, and, in particular to this case, constraints that are always slack, or constraints that can only be satisfied by a null solution vector. Once these are eliminated, the solution algorithm proceeds in earnest. Thus, it was only in the "middle ground" when there were roughly similar numbers of unconstrained and constrained futures that the problem was most challenging to solve.

The shape of the value function is similar to past shapes. The graph below compares the three curves for the T=1 time step (12 remaining time steps).

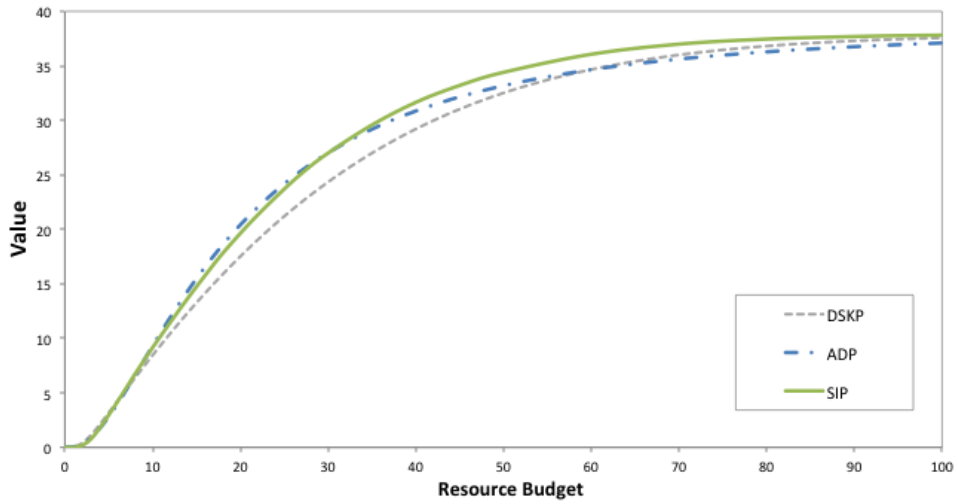

Figure 5-3 Comparing DSKP, ADP, and SIP Value Curves as a function of budget for T=1 on the 12 time step example problem

The curves are generally well aligned. The ADP and SIP implementations are slightly more aligned with each other than either is with the DSKP as the following matrix shows.

Table 5-5 Maximum absolute deviation between value functions for DSKP, ADP, & SIP approaches

Approaches	MaxAbsDev
DSKP vs ADP	3.00
DSKP vs SIP	2.65
ADP vs SIP	1.42

What might explain these differences? In the DSKP dynamic programming implementation, we necessarily have to discretize the random vector to generate the expected value. In the ADP implementation, we employ recursive regression machine learning to fit the parameters of a set of log-normal basis functions. However, this fit is done over a sample. Had we used a different random seed, the result would have been slightly different. In the SIP implementation, one structural difference is that we are approximating an N-stage problem with a two-stage formulation. This result is also the product of a sample, which though large enough to give us answers within the desired precision, would also give slightly different answers for a different random seed. Given these differences, we will interpret these curves as being closely aligned. We turn our attention now to solving the full scale problem.

5.3.2 Full Scale Problem Implementation

From the previous chapter the full-scale problem, which represents the system behavior we observed while at JIEDDO, employs the following data.

Full Scale Problem Data

T = 50 time periods

B = 2,500,000,000 resource units (but in units of millions)

p = 1/3 - likelihood of binary arrival in any given time step

$C \sim$ Log-normal(16.58, 3.57) distribution of cost given an arrival

$V \sim$ Uniform(0, 2C) distribution of value of an arrival given its cost

Recall our discussion in the previous chapter, where we recognized that while the resource budget is \$2.5B, that were we to operate our models in unit of dollars we are essentially saying that a decision could hinge on a single dollar. The least cost we recorded at JIEDDO – our case study for the WPSP – was \$100K. Thus, we scale our data in units of millions of dollars. Instead of rescaling the log-normal distribution, we employ the pragmatic short cut of re-scaling the sample.

Again we follow the same procedure of generating the random scenarios via a Monte Carlo simulation, dividing each element of the sample by \$1M, rounding the results to one significant digit, which would equate to \$100K, and then using this set as data for the SIP. The following table shows run times to solve single instances of the Full

Scale problem by increasing sizes of N. The SAA algorithm gives us a better solution and the means to determine the precision of our answer.

Table 5-6 Results of single instance of Full Scale Problem by increasing N

N Scenarios	ScenarioCost>Budget	%Constrained	Solution	Time(sec)
250	35	14.0%	1182.18	0.03
500	73	14.6%	1200.94	0.06
1,000	143	14.3%	1163.84	0.14
2,500	372	14.9%	1186.45	0.55
5,000	705	14.1%	1192.43	1.74
10,000	1,461	14.6%	1201.36	7.74
25,000	3,580	14.3%	1202.35	41.44
50,000	7,149	14.3%	1200.80	95.14
100,000	14,208	14.2%	1199.67	737.78

We employ a desired error bound of 1% of the budget, which translates to 25 resource units. Applying our rule of thumb for the stochastic gradient of using an order of magnitude increase in precision to protect against premature convergence, we use an epsilon of 2.5 resource units.

Our desire is to find an N that balances between SAA stop time and the precision of the solution. We run SAA ten times at each level of N in order to determine the average number of iterations and the average stop time. From the same ten runs we also get an average solution and a 95% confidence interval for the true mean. To give the

confidence interval context, we use the term "precision", which is the confidence as a percentage of the mean SIP solution. Since the solution to the SIP is itself a random variable, it gives us some notion of the increase in solution precision we get from employing additional scenarios.

Table 5-7 Results from 10 Runs of the Simplified SAA Algorithm for varying N

N	Mean It'ns	Mean Stop Time	Mean Sol'n	95%CI-HW	Precision
250	8.2	0.46	1203.24	63.99	5.3%
500	4.8	0.44	1197.62	40.27	3.4%
1000	5.3	1.09	1195.26	25.89	2.2%
2500	4.9	2.97	1196.46	16.65	1.4%
5000	3.7	5.62	1191.72	14.62	1.2%
10000	3.3	25.34	1196.19	11.11	0.9%
25000	2.8	116.51	1194.72	4.53	0.4%

The table reinforces our intuitions. The mean iterations required before stopping declined gradually while the SAA mean stop time increased exponentially. Similar to many sampling statistics, the change in the confidence interval is roughly inversely proportional to the root of the change in the sample size. The chart below illustrates the trade between precision and computation time.

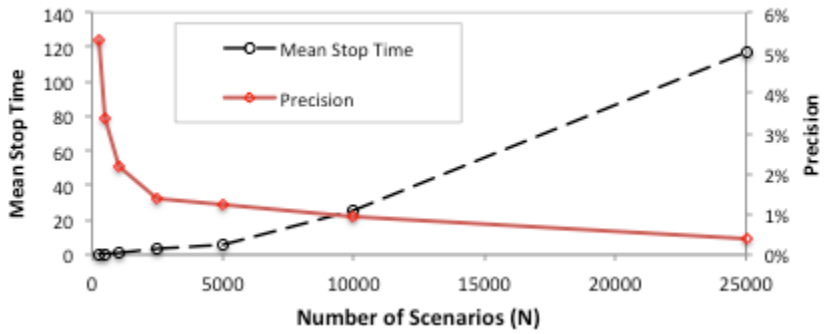

Figure 5-4 Illustrating the trade between computation time and precision for the Simplified SAA Application for the large scale problem for a range of scenarios N

In the quest for better answers we encounter again the issue of spurious precision. Recall that the values used come from a value model such as described in Chapter 3. While these models provide valuable information on the value of solutions, the answers they provide cannot be described as very precise since they reflect preference and choice. Thus as a rule of thumb, a process that generates mean solutions with a confidence interval of 1% of the mean is precise enough. The range of N between 5,000 and 10,000 highlights a potential high-payoff trade: precision increases by only 0.3% of the mean at a price of a five-fold increase in computation time.

We turn now to the generation of the $Q(x)$ curve as a function of available budget. In the ADP approach, this curve was a natural part of generating a solution. We had to explore the whole space to get that solution. In our 2-stage SIP approach we get that solution in two phases – now and future. However, there is a certain intrinsic value from understanding the shape of the curve. To generate the curve we need to repeatedly run

the model for gradually smaller budgets – in this case a sample of 34 budget levels evenly distributed between 0 and 2500. Of note, the computation time required to generate the curve was 1938 seconds.

The following figures compare $Q(x)$ to computation time and to the number of constrained future scenarios. Compared to the example problem data, computation time tracks closely with the number of constrained scenarios until the very end, when the resource budget so constrained most scenarios that in most scenarios the solution was a null vector.

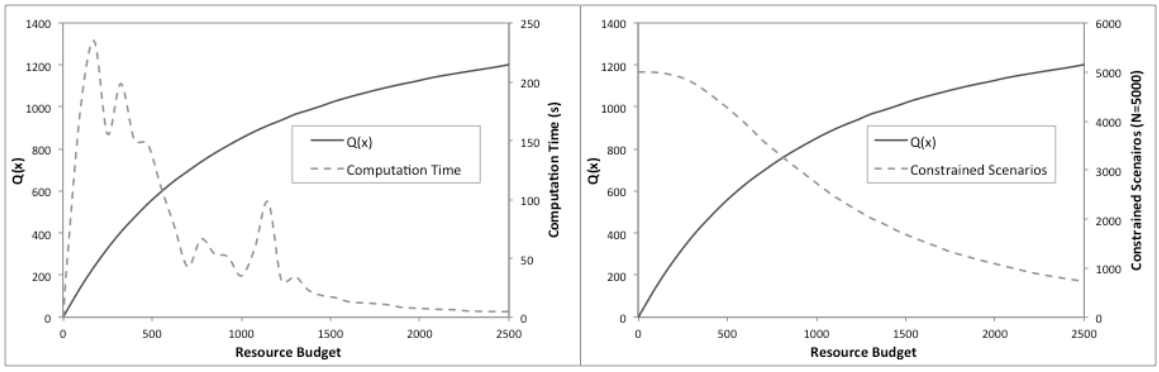

Figure 5-5 $Q(x)$ compared to solution times (left graph) and constrained futures (right graph)

Qualitatively the plot has changed. What happened with the interval of convexity for the initial part of the resource budget range? To understand this, we need to examine the effect of variance on the log-normal distribution. In our example problem, the log-

variance was 0.5. In the large problem the log-variance is 3.5. In the plot below, we show how the increase in variance affects the density plot of the log-normal.

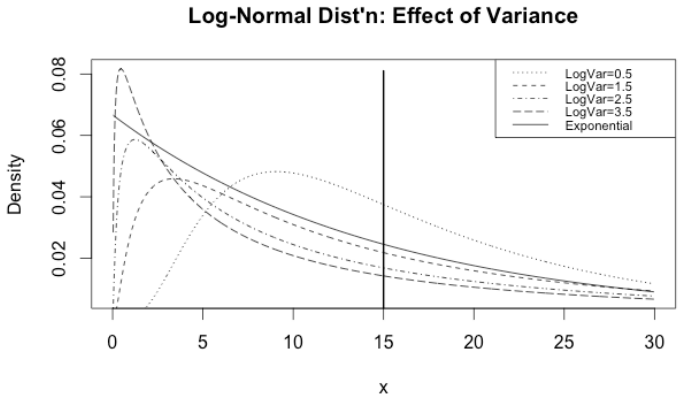

Figure 5-6 Depicting the effect of increasing log-variance on the log-normal distribution while holding the log-mean constant. The exponential with the same log-mean is shown as a point of comparison.

For comparison we plot an exponential curve. All the log-normal distributions have the same log-space mean: ln(15), while the exponential has a mean of 15. Note that as log-variance increases the mode of the distributions moves closer and closer to zero. For the log-variance of 3.5, which relates to our observed data, the period of increasing density is miniscule: as the table below shows, the mode is at x=0.45.

Table 5-8 Comparing Measures of Centrality for varying levels of Log-Variance

LogMean	LogVar	Mean	Median	Mode	95%-ile
2.71	**0.5**	19.26	15	9.10	48.23
2.71	**1.5**	31.76	15	3.35	111.58
2.71	**2.5**	52.36	15	1.23	201.73
2.71	**3.5**	86.32	15	0.45	325.03

The mode in the density distribution of the log-normal distribution of cost is what dictates the period of convexity in the value function. With the mode so close to zero, in the full scale problem the period of convexity is undetectable.

Our last view in this section is to compare output from our 2-stage SIP (2SSIP) to the output from the ADP solution. We reran our ADP approach using these data. Because the state space was bigger we used more runs, 100,000 versus the 12,000, in order to allow the learning algorithm more time to converge. This resulted in a slower run time of 1486 seconds – which is on an order with the 1938 seconds of computation time required for the SIP approach to generate the comparable curve. However, it took only a minute to solve the SIP for any particular budget-stage combination.

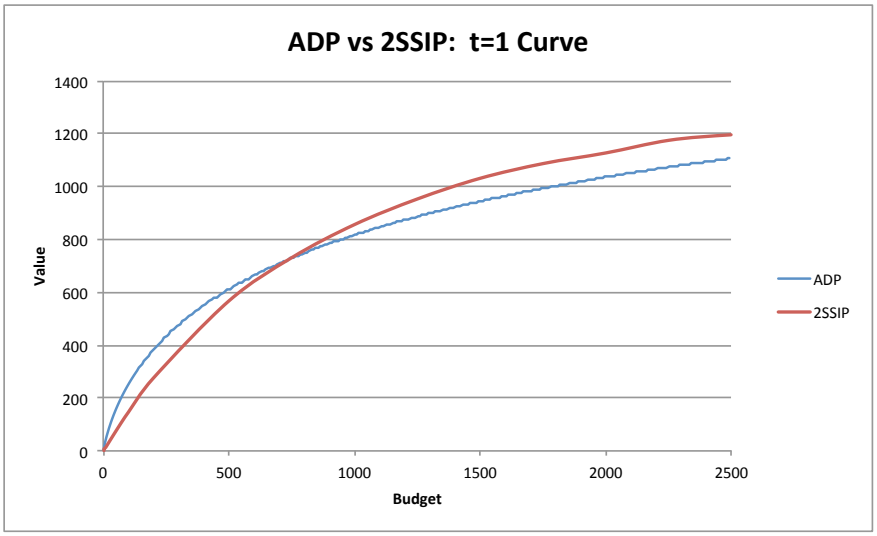

Figure 5-7 Comparing ADP to SIP value functions for the large scale problems

There are qualitative and quantitative differences in these curves. While there was relatively close alignment in the example problem, the differences here are more pronounced. Which curve to believe?

The ADP solution comes from a single run of the model. The value at t=1 with a full budget represents the expected value we can expect to accrue through the decision epoch at the start of the year. This value comes from a recursive regression fitting on a basis set of functions. In one run, it is trying to fit the entire state space trajectory. As we saw in Chapter 4, the quality of the fit is subject to the selection of the appropriate basis functions, which can involve some trial and error. The SIP implementation does not need to fit the values to a model – it yields an exact constrained sample mean for that point in the state space. Thus, our inclination is to trust the SIP solution.

5.4 Addressing Initiative Investment Options (First Stage Decisions)

The funding decision for a given initiative is not truly binary. An initiative may be acquired for different users in varying tactical scenarios. Examples in the counter-IED realm include route clearance, convoy operations, defenses for forwarding operating bases, and dismounted patrols. We might consider acquiring quantities just for the forces engaged, or only in the particular theater of war where most needed, or for the entire force. Thus, each initiative decision could actually represent a set of options.

We illustrate this principle with a system with two types of users and in different theaters of war: system A for user X in theater I, system A for user Y in theater I, system A for user X in theater II, and system A for user Y in theater II. In cases like these, it may improve the value of the decision to decompose each initiative decision into a set of binary options. Then the value and cost of each of these options must be estimated separately.

Occasionally, there may be a need to address interactions between initiatives. Two initiatives operating in the same tactical scenario may improve each other's operation through physical or operational synergies; in the negative of this effect, they may interfere with each other; or they may have independent effects but these must still be de-conflicted because they are operating in the same scenario and compete to defeat the same set of IEDs. In none of these cases can we treat the individual value of the options in these cases additively. We advocate that for each of the scenarios where these

interactions exist, the value and cost of each initiative *combination* be estimated separately.

For example, assume that the values of systems A and B in a particular theater or tactical domain vary based on whether the other system is present or not. Then we identify the value and cost of three cases: system A alone, system B alone, and system A and B. For the particular tactical domain, we can only choose at most one of these mutually exclusive combinations. This allows us to treat the values of any set of initiative options chosen as additive, since for scenarios with non-additive values we can only chose one option. Obviously, were we to have to many initiatives in the first stage that might be decomposed to these kind of mutually exclusive options, we might face a combinatorial explosion.

This ability to address nuances in the first-stage decisions is a modeling strength of the SIP approach. We show how the basic formulation might be modified to address interactions among first-stage initiatives.

5.4.1 Modified Formulation – Addressing Interactions

We revisit our previous deterministic equivalent formulation in order to describe how we might address interactions among the first-stage decisions. To refresh our recollection we show this formulation here again.

Deterministic Equivalent of the WPSP2SSIP

Maximize $\boldsymbol{Value^T Buy} + Q(\boldsymbol{Buy})$

Subject to $\boldsymbol{Cost^T Buy} \leq Budget$

Where

$Q(\boldsymbol{Buy}) = \mathbb{E}_S[Q(\boldsymbol{Buy}, s)]$

and

$Q(\boldsymbol{Buy}, s) = \max_{\boldsymbol{futBuy_s}}\{\boldsymbol{futValue_s}^T \boldsymbol{futBuy_s} | \boldsymbol{futCost_s}^T \boldsymbol{futBuy_s} \leq Budget - \boldsymbol{Cost^T Buy} \geq \mathbf{0}\}$.

We assume that all required initiative options and their mutually exclusive combinations have been defined. The data for these new options are all described in the first-stage data vectors ***Value*** and ***Cost.*** We note the new changes needed before presenting the modified formulation.

- Create the index *o* which identifies an option, *o* ε *O*.
- Create sets *Combination*(*K*), subsets of *O*. Each *K* relates to a particular theater or employment domain where we must consider interactions among the options in the first stage.
- Each *Combination*(*K*) which have as their members all the options *o* under a particular combination *K* that are judged to be mutually exclusive.
- We add a constraint specifying that for each *K* only one option can be chosen.

With these changes we obtain the following formulation.

Deterministic Equivalent of the WPSP2SSIP with Side Constraints

Maximize $\boldsymbol{Value^T Buy} + Q(\boldsymbol{Buy})$

Subject to $\boldsymbol{Cost^T Buy} \leq Budget$

$\sum Buy_{\boldsymbol{c}} \leq 1, \forall c \in Combination$(K)

Where

$Q(\boldsymbol{Buy}) = \mathbb{E}_S[Q(\boldsymbol{Buy}, s)]$

and

$Q(\boldsymbol{Buy}, s) = \max_{\boldsymbol{futBuy_s}}\{\boldsymbol{futValue_s}^T \boldsymbol{futBuy_s} | \boldsymbol{futCost_s}^T \boldsymbol{futBuy_s} \leq Budget - \boldsymbol{Cost^T Buy} \geq \boldsymbol{0}\}$.

5.5 An Attempt to Apply Bender's Decomposition WPSP

While we have shown that commercial solvers like Gurobi are readily able to solve useful sized instances of the WPSP, we describe here our attempt to apply Bender's Decomposition to solve WPSP. As the reader will see, the WPSP's structure does not allow for the generation of feasibility cuts and its optimality cuts do not yield strong bounds. The original L-shaped decomposition approach is owed to Bender (1962). For our presentation of the material we adapt from Freund (2004).

5.5.1 Special Structure of the WPSP Constraint Matrix

Allowing for K realizations of the random vector $\boldsymbol{\xi}$, each with probability α, the WPSP two-stage stochastic integer program can be expressed in the following expanded

form. Note that $\alpha^{\mathrm{T}}K = 1$. For compactness, we use $\boldsymbol{x}$ to indicate first stage buy decisions and $\boldsymbol{y}_\omega$ for the second stage binary decision for a given scenario ω, $\boldsymbol{c}$ is the vector of first stage costs, $\boldsymbol{v}$ is the vector of first stage values, and $\boldsymbol{fc}_\omega$ and $\boldsymbol{fv}_\omega$ are the vectors of second stage costs and values respectively for scenario ω.

$$
\begin{array}{lllllllll}
\max & \boldsymbol{v}^T\boldsymbol{x} + & \alpha\boldsymbol{fv}_1{}^T\boldsymbol{y}_1 + \alpha\boldsymbol{fv}_2{}^T\boldsymbol{y}_2 + \alpha\boldsymbol{fv}_3{}^T\boldsymbol{y}_3 + \ldots + \alpha\boldsymbol{fv}_k{}^T\boldsymbol{y}_k & & & & & & \\
\text{s.t.} & \boldsymbol{c}^T\boldsymbol{x} & & & & & & & \leq b \\
 & \boldsymbol{c}^T\boldsymbol{x} & + \boldsymbol{fc}_1{}^T\boldsymbol{y}_1 & & & & & & \leq b \\
 & \boldsymbol{c}^T\boldsymbol{x} & & + \boldsymbol{fc}_2{}^T\boldsymbol{y}_2 & & & & & \leq b \\
 & \boldsymbol{c}^T\boldsymbol{x} & & & + \boldsymbol{fc}_3{}^T\boldsymbol{y}_3 & & & & \leq b \\
 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
 & \boldsymbol{c}^T\boldsymbol{x} & & & & & & \boldsymbol{fc}_k{}^T\boldsymbol{y}_k & \leq b \\
 & \boldsymbol{x}, \boldsymbol{y}_\omega \in \{0,1\} & & & & & & &
\end{array}
$$

This "block-ladder" structure lends itself to an iterative solution approach known as Bender's decomposition. The WPSP in particular has a simplistic structure because each of the blocks is a single horizontal vector.

The basic approach is to iterate between a) solving an approximation of the problem made up of the first-stage constraints and cuts generated by a subset of the second-stage, and b) solving the sub-problems that represent the distinct second-stage realizations to in order to generate the cuts. In general, solving these sub-problems provides *feasibility* and *optimality cuts* that are then inserted as additional constraints in the approximation of the master problem.

The intent of this approach is to avoid solving the entire problem all at once, inserting only the constraints that might be active in the final basis. When the sub-problems no longer provide cuts, the problem is solved. As we shall see, only optimality cuts would apply to the WPSP.

5.5.2 Benders' Decomposition for Stochastic LP

We start by expressing the linear relaxation $WPSP_{LP}$ in the Deterministic Equivalent Program formulation. Note that while we have replaced the integrality constraint with an upper bound of 1 on the decision variables.

Deterministic Equivalent of $WPSP_{LP}$

Maximize $z = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{x} + Q(\boldsymbol{x})$

Subject to $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq b$

$\mathbf{1} \geq \boldsymbol{x} \geq \mathbf{0}$, $\boldsymbol{x}$ is 1 x n.

Where

$$Q(\boldsymbol{x}) = E_{\tilde{\xi}}[Q_{\omega}(\boldsymbol{x})]$$

And

$$Q_{\omega}(\boldsymbol{x}) = \max_{\boldsymbol{y}}\{\boldsymbol{f}\boldsymbol{v}_{\boldsymbol{\omega}}{}^{\mathrm{T}}\boldsymbol{y}_{\boldsymbol{\omega}} | \boldsymbol{f}\boldsymbol{c}_{\boldsymbol{\omega}}{}^{\mathrm{T}}\boldsymbol{y}_{\boldsymbol{\omega}} \leq b - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}, \mathbf{1} \geq \boldsymbol{y} \geq \mathbf{0}\}$$

Recall that for each realization ω of the random vector $\boldsymbol{\xi}$ there is the corresponding second-stage constraint $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{fc}_{\omega}\boldsymbol{y}_{\omega} \leq b$. As described previously, $\boldsymbol{fc}_{\omega}$ provides the cost data of the arrivals under a given scenario ω, while $\boldsymbol{fv}_{\omega}$ provides the value data for the same. For this scenario ω, we construct Sub-Problem ω (SP_{ω}).

$\mathbf{SP}_{\boldsymbol{\omega}}$

$$
\begin{aligned}
Q_{\omega}(x) = \quad & \max \quad \boldsymbol{fv}_{\omega}{}^{\mathrm{T}}\boldsymbol{y}_{\omega} \\
& \text{s.t.} \quad \boldsymbol{fc}_{\omega}{}^{\mathrm{T}}\boldsymbol{y}_{\omega} \leq b - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\
& \qquad \mathbf{1} \geq \boldsymbol{y}_{\omega} \geq \mathbf{0}
\end{aligned}
$$

We want to maximize the expected portfolio over the distribution of possible futures. The sub-problem restates this objective as it applies to a single instance ω of the possible futures. Note how the right hand side emphasizes that the knapsack capacity has been reduced as a result of first stage decisions. From the SP_{ω}, we define the dual problem DSP_{ω}, using $p_{\omega,1}$ as the scalar dual variable for the knapsack constraint and $p_{\omega,2}, \ldots, p_{\omega,n+1}$ as the vector dual variable for the upper bounds on $\boldsymbol{y}$.

$\mathbf{DSP}_{\omega}$

$$
\begin{aligned}
\Xi_{\omega} = \quad & \min \quad [(b - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}), \mathbf{1}^{\mathrm{T}}]\, \boldsymbol{p}_{\omega} \\
& \text{s.t.} \quad [\boldsymbol{fc}_{\omega} + \mathbf{I}_{n}]\, \boldsymbol{p}_{\omega} \geq \boldsymbol{fv}_{\omega} \\
& \qquad \boldsymbol{p}_{\omega} \geq 0
\end{aligned}
$$

For DSP_ω's feasible region, assume we care to enumerate all the feasible extreme points $\boldsymbol{p}_\omega^1,\ldots, \boldsymbol{p}_\omega^I$, and likewise all the extreme rays $\boldsymbol{r}_\omega^1,\ldots, \boldsymbol{r}_\omega^J$. Solving DSP_ω leads to two possible outcomes: either the problem is unbounded below and yields a feasibility cut, or there is an optimal solution to the problem, yielding an optimality cut.

If this dual problem is unbounded (primal infeasible), then the solution algorithm returns an extreme ray $\boldsymbol{r}_\omega^*$ with the property $[(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]\ \boldsymbol{r}_\omega^* < 0$, generating the corresponding feasibility cut for the is $[(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]\ \boldsymbol{r}_\omega^* \geq 0$. However, if we assume that $[(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]$ is always non-negative (cannot violate the entire budget using only 1st stage decisions), the second stage is always feasible.

Theorem 1: If we assume that $(b - \boldsymbol{c}^T\boldsymbol{x}) \geq 0$ then:

a) the $WPSP_{LP}$ has complete recourse,

b) Bender's Decomposition for the $WPSP_{LP}$ cannot generate feasibility cuts.

Proof: The $WPSP_{LP}$ will not have complete recourse if there exist $\{\boldsymbol{x}, \omega\}$ such that $K_2 = \{\boldsymbol{y}_\omega | \boldsymbol{fc}_\omega^T \boldsymbol{y}_\omega \leq b - \boldsymbol{c}^T\boldsymbol{x},\ \ \boldsymbol{y}_\omega \in (0,1)\} = \varnothing$. For any given ω, K_2 is only empty if some $\boldsymbol{x}$ was chosen in the first stage such that $\boldsymbol{c}^T\boldsymbol{x} > b$. We assumed that $b - \boldsymbol{c}^T\boldsymbol{x} \geq 0$, therefore there is some $\boldsymbol{y}_\omega$ satisfying K_2 – even if it is only zero – and there will be no infeasibilities from which to generate cuts.

We define the variable θ_ω which we will use as a surrogate for $Q_\omega(x)$. Solving the sub-problem DSPω yields a solution $\Xi_\omega = [(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]\, \boldsymbol{p}_\omega{}^*$. Note that by strong duality, $\Xi_\omega = Q_\omega(x)$. This will yield the optimality cut $[(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]\, \boldsymbol{p}_\omega{}^* \geq \theta_\omega$. Recall the vector of probabilities α for the K instances of the random vector $\boldsymbol{\xi}$. We can restate our problem in the following fashion, which we term the *Full Master Problem* (FMP), which we first express first with the cuts added to the Deterministic Equivalent, and then immediately below we convert the cuts to show the FMP in standard canonical form.

FMP

$$\max_x \quad z = \boldsymbol{v}^T\boldsymbol{x} + \boldsymbol{\alpha}^T\boldsymbol{\theta}$$

$$\text{s.t.} \quad \boldsymbol{c}^T\boldsymbol{x} \leq \boldsymbol{b}$$

$$[(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]\, \boldsymbol{p}_\omega{}^i \geq \theta_\omega;\ i = 1,\ldots, I;\ \omega = 1,\ldots,K$$

$$\boldsymbol{x} \geq \mathbf{0},\ \boldsymbol{\theta} \text{ unrestricted.}$$

FMP – Canonical Form

$$\max_x \quad z = \boldsymbol{v}^T\boldsymbol{x} + \boldsymbol{\alpha}^T\boldsymbol{\theta}$$

$$\text{s.t.} \quad \boldsymbol{c}^T\boldsymbol{x} \leq \boldsymbol{b}$$

$$\boldsymbol{c}^T\boldsymbol{x} p_{\omega l}{}^i + \theta_\omega \leq [b, \mathbf{1}^T]\, \boldsymbol{p}_\omega{}^i;\ i = 1,\ldots, I;\ \omega = 1,\ldots,K$$

$$\boldsymbol{x} \geq \mathbf{0},\ \boldsymbol{\theta} \text{ unrestricted.}$$

The FMP has replaced K decision variable vectors $\boldsymbol{y}_\omega$ with K scalar variables θ_ω. The trade however is a very large number of constraints – all the extreme points of the dual of the second stage. Bender's Decomposition solution approach is to work with a *Reduced Master Problem* (RMP), adding these constraints only as we need them. Consider an iteration of this approach, using j of the extreme points and k scenarios.

RMP

$$\begin{aligned} \max \quad & z^k = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\theta} \\ \text{s.t.} \quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \le b \\ & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}p_{\omega l}{}^{i} + \theta_\omega \le [b, \mathbf{1}^{\mathrm{T}}]\boldsymbol{p}_\omega{}^{i};\ i = 1,\ldots,j;\ \ \omega = 1,\ldots,k \\ & \boldsymbol{x} \ge 0,\ \boldsymbol{\theta} \text{ unrestricted} \end{aligned}$$

After solving this problem, we obtain a value z^k and a solution x^k. This value z^k is an upper bound on z from FMP, since we cannot hope to improve on the value obtained from RMP by adding more constraints. We check whether x^k violates any of the non-included constraints $[(b - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}), \mathbf{1}^{\mathrm{T}}]\boldsymbol{p}_\omega{}^{i} \ge \theta_\omega$, $i = j+1,\ldots, I$ by solving the K-k sub-problems DPω. If no new optimality constraints are generated, we are done.

This is the basic approach employed for stochastic linear programming. We next describe how we extend these concepts to the integer programming case.

5.5.3 Delayed Constraint Generation Algorithm for WPSP

We describe an adaptation of Bender's decomposition for the WPSP. Birge and Louveaux describe the Integer L-shaped Method wherein the master and sub-problem iteration is embedded within a branch and cut scheme. They describe three relaxations: the integer constraints for $\boldsymbol{x}$ in the first stage are relaxed in the branch and cut scheme, the feasibility constraints on $\boldsymbol{x}$ in the second stage are replaced with feasibility cuts, and the definition of the value function $Q(\boldsymbol{x})$ is modified to allow its approximation via a continuous decision variable $\boldsymbol{\theta}$.

In the WPSP, the first stage problem is generally small, frequently a single binary decision. The main issue is estimating the value of $Q(\boldsymbol{x})$. The decomposition allows us to solve individually k dual linear sub-problems, each yielding a corresponding set of k cuts on $\boldsymbol{\theta}$ which are returned to the reduced master integer program.

In the LP case, we had strong duality between primal and dual sub-problem solutions. In the integer case, this is no longer true, but the linear dual can still be used to provide optimality cuts. This does, however, introduce the question of when to stop the algorithm.

The random vector $\boldsymbol{\xi}$ in the WPSP is not finite; instead, in our approaches to date we have approximated our solution by using Monte Carlo sampling from $\boldsymbol{\xi}$. In our approach here, for each sample, we will generate optimality cuts via the linear dual sub-problems. We will then use the values of the sub-problems to provide an estimate of master-problem objective function and thereby a lower bound. The optimality cuts are

introduced and the reduced master problem is solved to generate an upper bound. When the optimality gap between the upper and lower bound is small enough, the algorithm stops.

WPSP Delayed Constraint Generation Algorithm

0. Initialize: OldSol = ∞, NewSol = 0, *CutSet* = Ø, $k = 0$.
1. Define RMP^k with *CutSet* from k scenarios.
   a. If *CutSet* not empty, solve $z^k = \max_{\boldsymbol{x} \in X} \boldsymbol{v}^T\boldsymbol{x}^k + \boldsymbol{\alpha}^T\boldsymbol{\theta}^k$. Else, set $\boldsymbol{x}^k = \mathbf{1}$.
   b. If *CutSet* not empty, OldSol <– z^k.
   c. Let $\boldsymbol{x}^k$ be the incumbent solution.
2. Generate new increment of K scenarios; $k = k + K$
3. For $\omega = 1,\ldots,K$, solve the sub-problems:
   a. $\Xi_\omega = \min\{[(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]\, \boldsymbol{p}_\omega, \text{ s.t. } [\boldsymbol{fc}_\omega + \mathbf{I}_n]\, \boldsymbol{p}_\omega \geq \boldsymbol{fv}_\omega, \boldsymbol{p}_\omega \geq 0\}$.
   b. Generate *CutSet* = $\{\theta_\omega \leq [(b - \boldsymbol{c}^T\boldsymbol{x}), \mathbf{1}^T]\boldsymbol{p}_\omega, \omega = 1,\ldots,k\}$
4. Set NewSol <– max{LB, $\boldsymbol{v}^T\boldsymbol{x}^k + \boldsymbol{\alpha}^T\Xi^k$}.
5. If abs(OldSol – NewSol) $\leq \varepsilon$, then stop. Else, add *CutSet* to RMP(k) and return to step 1.

5.5.4 Computational Results

We reconsider the dual sub-problem to see what structure can be exploited. Recall that each of these is the dual of a single constraint knapsack problem.

DSP$_\omega$

$$\Xi_\omega = \quad \min \quad [(b - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}), \mathbf{1}^{\mathrm{T}}]\, \boldsymbol{p}_\omega$$

$$\text{s.t.} \quad [\boldsymbol{fc}_\omega + \mathbf{I}_n]\, \boldsymbol{p}_\omega \geq \boldsymbol{fv}_\omega$$

$$\boldsymbol{p}_\omega \geq 0$$

There are two possibilities; (1) The knapsack constraint is taught, and we solve the DSB as a linear program via a solver, or (2) The constraint is slack, so we set $\boldsymbol{p}_\omega{}^{\mathrm{T}} = [0, \boldsymbol{fv}_\omega{}^{\mathrm{T}}]$; i.e., all the items will be acquired. The table below shows the results we obtained compared to the previously shown results from solving via Gurobi. Note that in general only 3 or 4 iterations are required. We used an epsilon of 0.5% of the budget.

Table 5-9 Bender Decomposition Approach Results Compared to Standard Gurobi Solver Approach

N Scenarios	Bender Itn's	Bender Solution	Bender Time (s)	Solver Solution	Solver Time (s)
500	4	1270.52	1.21	1182.18	0.03
1,000	4	1275.56	1.69	1200.94	0.06
2,500	5	1301.93	7.05	1163.84	0.14
5,000	4	1306.80	7.43	1186.45	0.55
10,000	4	1315.91	15.14	1192.43	1.74
25,000	3	1318.00	18.82	1201.36	7.74
50,000	3	1317.95	37.61	1202.35	41.44
100,000	3	1316.30	75.40	1200.80	95.14

The resulting algorithm runs initially slower than the SIP but Bender's run times grow at a slower pace, eventually catching up to the solver approach before 50,000 scenarios. The issue though is that the resulting solutions are biased high. The figure below provides a graphical comparison.

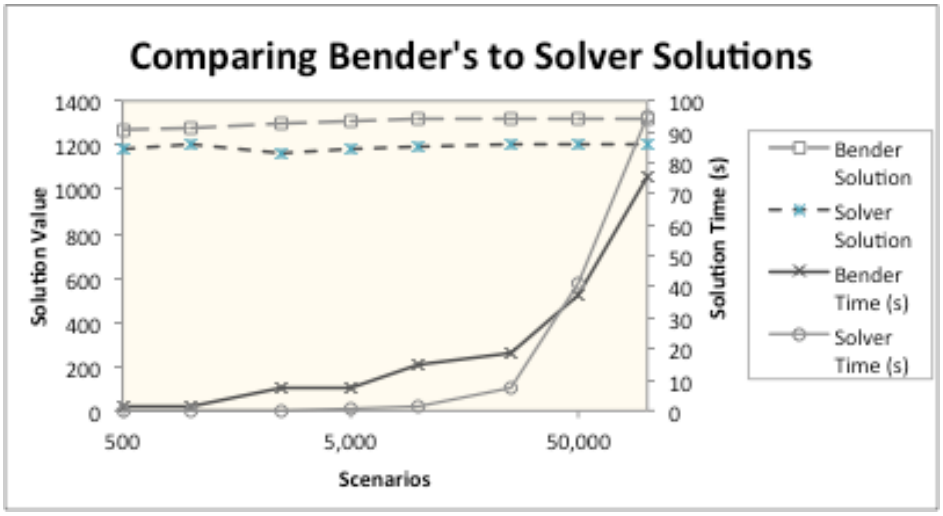

Figure 5-8 Comparing Bender's Decomposition Approach Results to those obtained via the Gurobi Solver

From the SIP and the ADP solutions, we know the solution should run around 1200 but Bender's yields solutions around 1315, about 10% high. The linearization of the binary knapsack with the high variance in the log-normal distribution, results in a

disparity between the binary solution and the continuous knapsack with an upper bound of 1. We illustrate with a simple example using the data shown earlier in the chapter.

Table 5-10 Example Scenario Data

Costs

Scenario\ Wks	1	2	3	45	46	47	48	49	50	51	52
1	0	0	1	4	6	0	0	0	161	0	38
2	8	0	0	0	310	0	1	0	0	0	0
3	0	0	0	4	0	0	0	0	18	293	0
4	0	50	4	0	0	10	13204	0	0	0	289
5	0	0	0	0	0	0	0	30	0	9	0

Values

Scenario\ Wks	1	2	3	45	46	47	48	49	50	51	52
1	0	0	0	1	1	0	0	0	289	0	58
2	3	0	0	0	425	0	3	0	0	0	0
3	0	0	0	5	0	0	0	0	9	180	0
4	0	58	8	0	0	2	15018	0	0	0	230
5	0	0	0	0	0	0	0	52	0	7	0

Consider scenario 1 of the example data. We will only use the visible data, items 1-3 and 45-52. If the available budget were 100, under the binary decision variable constraint we can only accept items 3, 45, 46, and 52. This only consumes 49 of the 100 resource units available and yields a value of 60. But if remove integrality and allow the amount of an item be any number between 0 and 1, the results are quite different. We would consume 0.62 of item 50, for a yield of 179.50. This is an extreme example, but it should give the reader some insight into the source of the gap.

There are strong parallels between Bender's decomposition and SAA algorithm. In both cases we were solving the SIP by increasing the overall sample size until we converged to a desired degree of precision. The attraction of the SAA algorithm is that it requires less customized coding and it yields the statistics necessary to create a confidence interval around the result.

5.6 Summary

In our exploration of stochastic integer programming approaches to the WPSP we have gained some insights about this method: its low barriers to implementation, its ease of use, and its modeling flexibility.

Implementing the WPSP2SSIP requires two basic technologies: the ability to generate scenarios via Monte Carlo, and a commercial solver able to handle reasonably sized integer programs. Some commercial packages now incorporate the ability to generate Monte Carlo simulations and solve the resulting stochastic program. The technologies are readily available to government organizations, most of which employ operations researchers, industrial engineers, and other mathematical scientists able to use implement this technology.

Once implemented, for the analyst it is a simple question to update the data from the most recent decision data, add the recent arrival(s) data, run the simulation to generate the constraint data, and then solve. With a little experimentation, it should be readily apparent which size problem – how many scenarios – solves in a timely fashion. Solving

a few times with different random samples informs the analyst as to the variability of the solution. The solution informs the decision maker as to whether to acquire or not. A simple parameterized sensitivity analysis can inform the decision make what increase/decrease in value or cost for the initiative might result in a change of solution.

Should decision makers wish to explore partitioning recent initiatives into *options* which could then be examined as combinations of options, the 2SSIP formulation adapts effortless to address this change. It also readily accepts other constraints such as satisficing strategic goals or imposing interaction constraints.

In the following chapter, we conclude this thesis by comparing the dynamic programming approaches to the stochastic integer programming approach, and consider further research.

CHAPTER 6 – CONTRIBUTION, FUTURE RESEARCH, AND CONCLUSIONS

6.1 Contributions

In this dissertation, we described a dynamic resource allocation problem faced by governments during longer conflicts. During the fluid, urgent needs of an enduring conflict, new war-fighting solutions present themselves over time randomly. Which of these should they choose and which should they decline? Using the Joint Improvised Explosive Device Defeat Organization (JIEDDO) as a motivational example for this problem, we developed solution methods to two sides of this particular problem.

- Military solutions are not fungible – their value lies in the utility they provide the using government across a variety of functional domains. Yet they ultimately compete for the same dollars. To address this need, we developed a decision analytic approach for how decision makers might put on a common scale of value different types of solutions, arriving sequentially over time, coming to fruition at different points in the future, all with varying likelihood of success. Via this methodology, using their own values, beliefs about the future, and discounting preferences, decision makers can compare disparate initiatives on a common scale of value.
- Given a means to assign value to arriving solutions, and some history of past arrivals, it may not be clear at a given point in the execution of an annual budget

what is a good investment and what is a bad one. This sequential decision-making naturally fits the type of challenge addressed by dynamic programming, but the need to solve the problem reasonably fast led us to develop an approximate dynamic programming. This approach provides fast approximate solutions.

- In parallel, we also developed a stochastic programming approach to the same problem. We demonstrated for a set of different example problems how the two approaches yield similar answers for the same data. This stochastic programming approach can also provide fast approximate solutions, but with more time can provide more precise solutions, and provides greater modeling flexibility.

Given that this thesis dealt with three separate intellectual thrusts - decision analysis, dynamic programming, and stochastic programming - we reflect on how they might all function in concert and the implications for future research.

6.2 Lessons Learned

6.2.1 Value functions and their use in Military Planning

Chapter 3 described a decision analytic approach measuring the military value of initiatives in a wartime portfolio acquisition context. This measure approach, Discounted Expected Net Initiative Value (DENIV), employed several concepts. First, an initiative's value is fundamentally measured in the context of the current portfolio: the net change in portfolio value resulting from adding the initiative. The net change in value must be modified by its likelihood of being realized, and discounted by the time expected until it

may be realized. Having this method in place allowed us to consider numerical optimization since we now had benefit and cost data for initiatives.

DENIV was heavily influenced by our experiences with JIEDDO, our inspirational case study. Implementing DENIV requires that the defense agency have in a place a portfolio value model and that they update the model regularly. With this process in place, the determination of an initiative's net contribution to the portfolio becomes feasible. It is our belief that this process was done *qualitatively* in the mind of the decision makers. Taking the additional steps to convert this qualitative understanding of the portfolio into a quantitative understanding would not just enable DENIV but provide a clearer context for the strategic identification of shortfalls and their prioritization in other venues such as science and technology investment.

6.2.2 Other Applications in Sequential Decision Making

The DENIV model offered an approach for addressing sequential valuation of non-fungible items arriving at random intervals to a decision maker. This is a similar situation to many entities that engage in research and development, where there is an urgency to bring new solutions to market, where there is a sequential nature to decision making, and where there is some element of preference in the portfolio beyond the measure of future profit.

For these kinds of entities, portfolio thinking is a valuable first step. Describe the desired theoretical portfolio, and what are the features of this desired portfolio. Contrast that with the as-is portfolio. The value of any new addition to the portfolio should be

viewed as the net improvement this would create between the as-is and the desired state. Because this value has not been realized, one should realistically consider both the likelihood and the time frame of its realization; that is, one should employ the discounted expectation of value. Many organizations develop solutions with similar time frames and similar likelihoods, and therefore may by force of habit overlook these considerations. Using discounted expected net value may ensure that solutions they may actually value do not get overlooked.

6.2.3 Lessons Learned

Case Studies

Development of this approach would benefit from a strong set of case studies that will help clarify many issues: in particular the issues of discounting and accounting for the likelihood of success. While discounting is common when the metric is monetary units, discounting a normalized measure of value is not. Furthermore, in this paper we described a situation where decision makers would discount at different rates for different kinds of solutions. In general what aspects of preference might lead decision makers to discount at different rates? Some solution types will naturally take longer to bring to fruition than others. While one might be able to develop useable software in a few months, warships take years to build. Likewise, the best pharmaceuticals still require years to go from initial lab results to general use. It would make sense then that

discounting should then take into account a "normal" wait time for a solution of certain types. The "right" way to do this will likely take case studies and experimentation.

Risk Preference and Utility

While expectation is a natural way of addressing uncertainty, in wartime decision makers may not be risk-neutral. Might not a utility-based approach – one that takes decision makers risk preferences into account - be more appropriate? In the example we provided in this paper, we only provided binary results: success or failure. Furthermore the probability of success was elicited from subject matter experts. A richer more informative approach may come via Monte Carlo simulation of project schedules (for an example, see McCabe, 2003). This kind of approach can provide a joint distribution of success, time to completion, and cost. In this context, one could envision an integrated approach to addressing risk, utility and discounting.

6.2.4 Implications for Optimization: Are initiative values proportional to their costs?

In Chapters 4 and 5 we considered two different approaches to optimization. While JIEDDO acquisition history provided the insight that initiative costs were log-normally distributed, there was no history of the value of initiatives. Thus, we had to make a critical assumption about the value of initiatives: that the value of an arriving initiative was randomly distributed around a mean proportional to the arrival's cost. Since costs were log-normally distributed, then by this assumption the values should be

similarly distributed. If the values were not in some sense proportional to cost, it might easily happen that the very expensive solutions are never acquired because their costs far exceed their value.

This assumption seemed perfectly rational in the context of an optimization study using randomly generated data. A real question remains how well that assumption might hold once DENIV were employed. Because DENIV measures value primarily as a net change in portfolio, only a few initiatives might have the ability to truly make an impact on the portfolio. It may be then that a system like DENIV results in a few very valuable initiatives and lots of low impact, but necessary, initiatives; a result which may parallel their cost distribution. But this is only speculation. This is a clear area for further research.

6.3 Optimization Approaches to WPSP

6.3.1 Comparing the two approaches

We examined these two different approaches by first developing specific quantitative approaches, testing these via numerical examples using a "small" problem and then extended from that to the large problem. A key insight we gained into the nature of this problem with both approaches is that what made the problem "big" was not so much the log-mean of the cost distribution but its log-variance. Rescaling the distribution changed the log-mean but not the log-variance.

Approximate Dynamic Programming

The approximate dynamic programming (ADP) yielded fast solutions: run times ranged from 10 seconds to a few minutes depending upon how many iterations of the problem were desired and the number of time steps in the decision epoch. The solutions compared favorably to the dynamic programming method, but yielded solutions in the worst case we examined that were thousands times faster.

However, the value curve was the result of extensive effort to fit a cumulative log-normal distribution of unknown parameters via recursive regression. While we developed a workable approach to doing this, whether this method was the best way to do so is open to debate. If we consider merely the value of the solution at the boundary, then this approach yielded workable solutions.

In the absence of any other viable approaches, the extensive coding, experimenting, and tuning required to develop this approach might be worth it. Indeed, to paraphrase Powell, ADP offers solutions to problems where no other solution method exists. However, as we saw for the WPSP, there exists another approach.

Stochastic Integer Programming

The stochastic integer programming (SIP) approach to the WPSP also offered fast approximate solutions with small Monte Carlo samples. Since the solution of the SIP-WPSP is itself a random variable, greater precision could be obtained by generating a

larger sample of futures. However, pursuit of greater precision might be a quixotic quest since the problem's input data is imprecise.

Conversely, the need for incorporating rare event information translates to a need for as large a sample from the random vector ξ as possible. The Sample Average Approximation (SAA) methodology proved a valuable complement to the SIP approach in this regard. Compared to solving an instance of the SIP with N samples from the random vector ξ, the SAA methodology allowed the same order of sampling via a solving a sequence of SIP using smaller samples. Sampling this way yielded not just an optimal expected value but also a measure of the precision of the answer via the variance the sequence of solutions.

Comparing the Approaches

Compared to ADP, defense organizations may find SIP much simpler to implement: the problem data can be generated easily from spreadsheets and the formulation is relatively simple to implement for individuals with an exposure to mathematical programming. Lastly, the SIP offers inherent modeling flexibility not present in the ADP approach. The SIP implementation can easily handle common scenarios such as multiple arrivals in the same week, decomposing arrivals into multiple dependent options, or the layering on of strategic priorities. It can easily handle changes in the assumptions about the random distribution of arrivals – changes which could lead to changes in the structural implementation of the ADP approach.

The fundamental difference between the two approaches is that in ADP the decision logic is embedded in the simulation framework. Decisions are made in the context of the simulation, and via recursive regression, we "learn" the value function, for any given state. The recursive regression requires an understanding of good basis functions. In SIP, the decision logic is orthogonal to the simulation framework. If something changes regarding how we perceive the future, we can change the simulation framework but preserve the decision logic.

6.3.2 Future Research

Multi-Stage Decision Making

The optimization approach employed here has obvious applications to the other sequential decision making entities such as the pharmaceutical industry. In this problem we focused on the single phase decision, which is more a worst-case scenario. In practice, both in pharmaceuticals and as we saw in JIEDDO, development of sequential projects is done in stages with decision points along the way. Keles and Hartman (2007) use ADP to explore decision making in multi-stage R&D pharmaceutical context. Their assumptions made projects, while stochastic, rather uniform in their randomness compared to the very large range in data variance we observed at JIEDDO. While we found the single stage problem was more easily and flexibly modeled via SIP, this multi-stage context may prove more amenable to modeling via ADP.

Fitting Log-Normal Cumulative Curves Recursively

One challenge for ADP in particular was how to best recursively fit the parameters of a log-normal cumulative distribution. This was a challenging problem, since fitting this curve essentially dictated the value curve solution of the ADP. This by itself would likely make a useful journal paper.

Using Sample Average Approximation as a Sensitivity Analysis Tool

The SAA methodology employed a simple gradient function stopping criteria. Another methodology that has its roots in discrete event simulation is Optimal Computing Budget Allocation (OCBA) (Chen, 1995). This approach is used to optimize the use of computing time in a simulation. The idea is for the simulation to generate only the number of samples needed to support the decision. If a choice has to be made between two random variables, the simulation uses a sequence of samples to determine which variable is superior to within a desired statistical confidence level.

Linking this approach to SAA may yield a fast and innovative approach for conducting dynamic sensitivity analysis for SIP. For a single instance of the SIP, assume a result $\boldsymbol{x}$. A decision maker may want to know how sensitive the result is to input parameters. One approach might be to conduct a parameterized study of the inputs, where the SAA method is blended with OCBA logic to run only enough SIP iterations as needed to determine the sensitivity of the solution $\boldsymbol{x}$.

6.4 Conclusion

Sequential decision making surrounds us. Sequential decision making is a natural situation that arises in competitive environments where decision makers may not have time to wait to perform holistic decisions. Instead, the decisions must be made as the decision opportunities present themselves. Christmas shoppers buy on Black Friday hoping better deals do not arrive before Christmas. In tight real estate markets, buyers cannot simply sleep on it because the attractive house may be gone by the time they come back. Employers often need to fill vacancies quickly and must balance between "best qualified" and "first come, first served".

Our focus in this paper was on wartime acquisition decision making: how to best acquire wartime capabilities sequentially. As these words are written the United States has been at war, in one fashion or another, for more than thirteen years, with new threats looming. Hopefully, the application of quantitative and qualitative analytic methods will contribute to making the world a safer place and might also be applied to a variety of other endeavors where the decisions made today can have significant impact on the long-term profitability, efficiency or well-being of an organization.

APPENDIX A. WPSP DYNAMIC PROGRAMMING CODE

As described in the main body of this dissertation, we employed the statistical programming language R for all the code in this dissertation. The DP code employs the recursive logic described by Papastavrou, Rajagopalan, Kleywegt (1996).

```
start.time <- proc.time()  #initializes timing of the loop to give us a run time measure.

Capacity <- 100
Deadline <- 12

p <- 1/3
minWt <- 1 #not truly the min weight - but the "observed min"
maxWt <- 50 #not truly the max weight – used initially to characterize the distribution
meanLogNorm <-2 #log(maxWt*minWt)/2
varLogNorm <- 2.5
rate <- 1/exp(meanLogNorm + .5*varLogNorm)
sdLogNorm <- varLogNorm^(1/2)

set.seed(501)
totalSamples <- 5000000
lnDenDraw <- array(round(rlnorm(totalSamples,meanLogNorm,sdLogNorm),0))
maxWt <- max(lnDenDraw)

numRwdCases <- 20
EV <- array(0,c(Deadline+1,Capacity+1))
R <- array(2*maxWt,c(Deadline+1,Capacity+1,maxWt+1))
rwdLevels <- array(0,c(maxWt+1,numRwdCases))
rwdSteps <- array(0,maxWt+1)
lnDens <-array(0,maxWt+1)
mass <- array(0, maxWt+1)


tb<-table(lnDenDraw)
```

```
freqs <- tb/totalSamples

if (min(lnDenDraw)==0){
  for (i in 0:maxWt) {
    if (i+1>dim(freqs)) {
      lnDens[i+1] <- 0
    } else {
      lnDens[i+1] <- freqs[i+1]
    }
  }
} else {
  lnDens[1] <-0
  for (i in 1:maxWt) {
    if (i+1>dim(freqs)) {
      lnDens[i+1] <- 0
    } else {
      lnDens[i+1] <- freqs[i]
    }
  }
}

ruDens <- 1/(numRwdCases)

for (w in 0:maxWt) {
  rwdSteps[w+1]<-2*w/(numRwdCases-1)
  rwdLevels[w+1,1]<-0
  for (r in 2:numRwdCases) {
    rwdLevels[w+1,r]<-rwdLevels[w+1,r-1]+rwdSteps[w+1]
  }
}

# make the join mass distribution
mass <- lnDens*ruDens

# print(ruDens)
# print(mass)
for (t in Deadline:1) {
  cat("\nt = ",t,"\n")
  flush.console()
            for (c in Capacity:0) {
                        for (w in 0:maxWt) {
                                    for (r in 1:numRwdCases) {
                                                if (w <= c) {
                                                            R[t,c+1,w+1] <- EV[t+1,c+1] - EV[t+1,c+1-w]
                                                            if (rwdLevels[w+1,r] <= R[t,c+1,w+1]) {
                                                                        EV[t,c+1] <- EV[t,c+1] + p*EV[t+1,c+1]*mass[w+1]
                                                            } else {
                                                                        EV[t,c+1] <- EV[t,c+1] + p*(rwdLevels[w+1,r] +
```

```
                                                                EV[t+1,c+1-w])*mass[w+1]
                                                }
                                        } else {
                                                EV[t,c+1] <- EV[t,c+1] + p*EV[t+1,c+1]*mass[w+1]
                                        }
                                }
                        }
                        EV[t,c+1] <- EV[t,c+1] + (1-p)*EV[t+1,c+1]
                }
}
end.start<- proc.time()-start.time
print(c("Runtime=",end.start))

# Plotting results
maxTime<-Deadline+1
timeRng <- 1:maxTime
x <- 0:Capacity
qtrStep <- round(maxTime/4)
#qtrStep
midStep <- round(maxTime/2)
threeQtrStep <- round(3*maxTime/4)
#midStep
nearEnd <- Deadline
qtrWt <- wtLevels[max(1,round(numWtCases/4))]
qtrWt
midWt <- wtLevels[max(2,round(numWtCases/2))]
midWt
twoThirdsWt <- wtLevels[max(3,round(numWtCases*2/3))]
twoThirdsWt
yRange <- max(EV)

maxCap <- Capacity+1
capRng <- 1:maxCap
outPuts <- EV[,capRng]
capRng <-1:Capacity
plot(main=paste("DSKP DP (CostMean:",meanLogNorm,", CostVar:", varLogNorm,")"), x, outPuts[1,],
ylim = c(0,1.2*max(EV)), xlab="State b", ylab="f()", type="l")
lines(x,outPuts[qtrStep,],lty="dashed")
lines(x,outPuts[midStep,],lty="dotdash")
lines(x,outPuts[threeQtrStep,],lty="dotted")
lines(x,outPuts[nearEnd,],lty="longdash")
legend("topleft",c("t=1",qtrStep,midStep,threeQtrStep,nearEnd),lty=c("solid","dashed","dotdash",
        "dotted","longdash"),cex=0.8)

nearEnd <- 1
tenthStep <- ceiling(Capacity/10)
qtrStep <- ceiling(Capacity/4)
midStep <- ceiling(Capacity/2)
```

```
timeRng <-1:maxTime
plot(main="DSKP DP:  f() vs Stage t by State b",timeRng,EV[,maxCap],ylim = c(0,1.1*max(EV)),
        xlab=paste("Stage t"), ylab="f()",type="l")
lines(timeRng,EV[,midStep+1],lty="dotdash")
lines(timeRng,EV[,qtrStep+1],lty="dashed")
lines(timeRng,EV[,tenthStep+1],lty="dotted")
lines(timeRng,EV[,nearEnd+1],lty="longdash")
legend("topright",c(paste("b=",Capacity),midStep,qtrStep,tenthStep,nearEnd),lty=c("solid", "dotdash",
        "dashed", "dotted","longdash"),cex=0.7)

tt <- floor(Deadline/2)
x <- 0:Capacity
qtrStep <- round(Capacity*.2)
midStep <- round(Capacity*.4)
threeQtrStep <- round(Capacity*.6)
nearEnd <- 1
maxCap <- Capacity+1
capRng <- 1:maxCap
outPuts <- R[tt,capRng,]

plot(main=paste("CritRwd vs State b by Cost c at t=", tt), x, utPuts[ Capacity*.8], ylim=c(0,max(EV)),
        xlab="State b", lab="CritRwd",type="l")
lines(x,outPuts[,qtrStep],lty="dotdash")
lines(x,outPuts[,midStep],lty="dotted")
lines(x,outPuts[,threeQtrStep],lty="longdash")
legend("topright",c(paste("cost=",Capacity*.8), threeQtrStep, midStep, qtrStep),
lty=c("solid", "dotdash", "dotted", "longdash"), cex=0.7)

tt <- floor(3*Deadline/4)
outPuts <- R[tt,capRng,]
plot(main=paste("CritRwd vs Capacity by Wt at t=", t), x, outPuts[,maxWt], ylim=c(0,max(R)),
        xlab="Capacity", ylab="CritRwd",type="l")
lines(x,outPuts[,qtrStep],lty="dotdash")
lines(x,outPuts[,midStep],lty="dotted")
lines(x,outPuts[,threeQtrStep],lty="longdash")
legend("topright",c(paste("wt=",maxWt),qtrStep,midStep,threeQtrStep),
lty=c("solid","dotdash","dotted","longdash"),cex=0.7)

max(EV)
write.csv(EV, "EVdefaultCase12.csv")
fileName <- paste("DSKP_startValue_logMean_",meanLogNorm,"_logVar_",varLogNorm,".txt", sep = "")
write(outPuts[1,],file=fileName,ncolumns=1)
outPuts[1,Capacity] # print out the expected value for the full budget at t=1
```

APPENDIX B. WPSP APROXIMATE DYNAMIC PROGRAMMING CODE

As described in the main body of this dissertation, we employed the statistical programming language R for all the code in this dissertation. The ADP code employs the described the double pass algorithm found in Powell (2007), the basis functions, step-size scheme, and recursive regression all as described in Chapter 4.

```
# ADP WPSP Double pass algorithm - uses a set of log-normal basis functions in recursive regression scheme to approximate the value function
# Programmed by Ronald F. A. Woodaman, this version dated 10 June 14, as part of GMU SEOR dissertation research

start.time <- proc.time()  #initializes timing of the loop to give us a run time measure.
#problem features

#general attributes
scaler <- 1000000
Deadline <- 50
p <- 1/3
Capacity <- 2500000000/scaler
maxN <- 100000#12000
randomize <- 1 #starts at different levels of budget in order to populate value space
seed <- round(301) #controling the random seed to ensure replicability of results
plotPoints <-300
plotStep <- ceiling(Capacity/plotPoints) #controlling density of plot

#cost/value distribution
meanLogNorm <- 16.58-log(scaler)
varLogNorm <- 3.57
sdLogNorm <- varLogNorm^(1/2)

#regression basis setup
meanArrival <- exp(meanLogNorm + .5*varLogNorm)
meanTotalArrival <- meanArrival*p*Deadline
```

```
meanTotalLogNorm <-log(meanTotalArrival)
rate <- 1/meanArrival
numTerms <- 4#round(Deadline/6,0) + 1
startPoint <- 0
secondVarianceFactor <- 2
termVariances <- 2
stepTuner <- .5^(.5)
startTuner <- 1
termStart <- startTuner*(meanLogNorm + .5*varLogNorm)
termStep <- sdLogNorm*stepTuner#(meanTotalLogNorm - meanLogNorm -
.5*varLogNorm)/(numTerms+1)
term <- (1:numTerms-startPoint)*termStep + termStart
termSD <- sdLogNorm
termSDtwo <- sdLogNorm*secondVarianceFactor^0.5
theta <- array(0,c(Deadline+1,numTerms)) #coefficient for linear function
if (termVariances > 1) thetaTwo <- array(0,c(Deadline+1,numTerms)) #coefficient for linear function
phi <- array(0,c(numTerms))
if (termVariances > 1) phiTwo <- array(0,c(numTerms))
print(paste("Mean Arrival Cost: ",format(meanArrival, nsmall=2,big.mark=",")))

#learning parameters
alpha <- 0.001
tuneUpPoint <- 1/5
shrinkPoint <- 3/4
shrinker <- 0.9999
infintesimal <- 0.001
ssqeAlpha <- 0.01
ssqe <- array(0,c(Deadline+1,maxN))
stochErr <- array(0,Deadline+1)

#turns on automated debugging features
deBugging <- 1
if (maxN>10) {
  deBugging <-0
}

v <- array(0,c(Deadline+1,maxN)) #array(0,Deadline+1)


#start loop
for (n in 1:maxN) {
  #display progress
  percentDone <- 100*n/maxN
     if (percentDone%%10==0) {
       print(paste("Iteration: ",n,", ",percentDone," percent Done"))
       flush.console()
     }
```

```
 if (randomize >0) {
  budget <- max(1,round((1-runif(1)^3)*Capacity,0))
  #startTime <-max(1,round((runif(1)^3)*Deadline,0))
  startTime <-1
 } else {
  budget <- Capacity
  startTime <-1
 }
 arrival <- array(0, Deadline)
 cost <- array(0,Deadline)
 reward <- array(0,Deadline)
 buy <- array(0,Deadline)


 if (deBugging > 0) print(paste("Starting n: ",n,"budget: ",budget,"startTime: ",startTime))

 for (step in startTime:Deadline) {
  arrival[step] <- (runif(1) < p)
  if (arrival[step]) {
   cost[step] <- round(rlnorm(1,meanLogNorm,sdLogNorm))
   reward[step]<- runif(1,0,2*cost[step])
   if (cost[step]<=budget) {
    budgetAfterbuy <- budget - cost[step]
    oldValueIfbuy <- sum(theta[step+1]*plnorm(budgetAfterbuy+infintesimal,term,termSD)) +
sum(thetaTwo[step+1]*plnorm(budgetAfterbuy+infintesimal,term,termSDtwo))
    oldValueIfpass <- sum(theta[step+1]*plnorm(budget+infintesimal,term,termSD)) +
sum(thetaTwo[step+1]*plnorm(budget+infintesimal,term,termSDtwo))
    delT <- oldValueIfbuy - oldValueIfpass
    if (reward[step]>delT) {
     buy[step]<-TRUE
     budget <- budgetAfterbuy
    }
   }
  }
  if (deBugging > 0) print(paste("After step=",step,"Arrival
Data:",arrival[step],cost[step],reward[step],"Action: ",buy[step],"New Budget: ",budget))

 }
 if (deBugging > 0) print(paste("Starting backward pass with budget:",budget))
 for (step in Deadline:startTime) {
  rise <- reward[step]*buy[step]
  run <- cost[step]*buy[step]
  v[step,n] <- rise + v[step+1,n]
  nextBudget <- run + budget
  if (deBugging > 0) print(paste("Step: ",step,", Buy: ",buy[step],", v: ",v[step,n],", budget: ",budget,",
nextBudget: ",nextBudget))
```

```
   #Recursive Regression
   oldTheta <- theta[step,]
   if (termVariances > 1) oldThetaTwo <- thetaTwo[step,]
   oldRegVal <- sum(oldTheta*plnorm(budget+infintesimal,term,termSD)) +
sum(oldThetaTwo*plnorm(budget+infintesimal,term,termSDtwo))
   stochErr[step] <- oldRegVal - v[step,n]

   if (n>1) {
    ssqe[step,n] <- ssqeAlpha*stochErr[step]^2 + (1-ssqeAlpha)*ssqe[step,n-1]
   } else {
    ssqe[step,n] = stochErr[step]^2
   }

   if (n>maxN*shrinkPoint) {
    alpha <- alpha*shrinker
   }

   phi <- plnorm(budget+infintesimal,term,termSD)
   phiTwo <- plnorm(budget+infintesimal,term,termSDtwo)
   theta[step,] <- oldTheta - alpha*(stochErr[step])*(phi)
   thetaTwo[step,] <- oldThetaTwo - alpha*(stochErr[step])*(phiTwo)

   if (deBugging > 0) print(paste("StochErr: ",stochErr[step],"Phi: ",phi))
   if (deBugging > 0) print(paste("Theta: ",theta[step,]))

   budget <- nextBudget
  }
}



capRng <- seq(0, Capacity, plotStep)
qtrStep <- round(Deadline/4)
midStep <- round(Deadline/2)
threeQtrStep <- round(3*Deadline/4)
nearEnd <- Deadline
t1curve <- array(0,c(2,Capacity))

nearEndValue <- array(0,length(capRng))
threeQtrStepValue <- array(0,length(capRng))
midStepValue <- array(0,length(capRng))
qtrStepValue <- array(0,length(capRng))
startValue <- array(0,length(capRng))

for (cap in capRng) {
  nearEndValue[cap/plotStep+1] <- sum(theta[nearEnd,]*plnorm(cap,term,termSD)) +
            sum(thetaTwo[nearEnd,]*plnorm(cap,term,termSDtwo))
  threeQtrStepValue[cap/plotStep+1] <- sum(theta[threeQtrStep,]*plnorm(cap,term,termSD)) +
```

```
            sum(thetaTwo[threeQtrStep,]*plnorm(cap,term,termSDtwo))
  midStepValue[cap/plotStep+1] <- sum(theta[midStep,]*plnorm(cap,term,termSD)) +
            sum(thetaTwo[midStep,]*plnorm(cap,term,termSDtwo))
  qtrStepValue[cap/plotStep+1] <- sum(theta[qtrStep,]*plnorm(cap,term,termSD)) +
            sum(thetaTwo[qtrStep,]*plnorm(cap,term,termSDtwo))
  startValue[cap/plotStep+1] <- sum(theta[1,]*plnorm(cap,term,termSD)) +
            sum(thetaTwo[1,]*plnorm(cap,term,termSDtwo))
  t1curve[1,cap/plotStep+1] <- cap
  t1curve[2,cap/plotStep+1] <- startValue[cap/plotStep+1]
  #print(paste("Cap: ",cap,",startValue: ",startValue[cap]))
}

plot(main=paste("ADP (CostMean: ",meanLogNorm,", CostVar: ",varLogNorm,", p:
1/3",")",sep=""),capRng,startValue,ylim = c(0,1.2*max(startValue)),xlab=paste("State b
(N=",maxN,")"),ylab="Value",type="l")
lines(capRng,qtrStepValue,lty="dashed")
lines(capRng,midStepValue,lty="dotdash")
lines(capRng,threeQtrStepValue,lty="dotted")
lines(capRng,nearEndValue,lty="longdash")
legend("topleft",c("t=1",qtrStep,midStep,threeQtrStep,
nearEnd),lty=c("solid","dashed","dotdash","dotted","longdash"), cex = 0.7)

#print(paste("Basis Fns#: ",numTerms,", mean(ssqe): ",format(mean(ssqe), nsmall=2,big.mark=",")))

end.start<- proc.time()-start.time
print(end.start)
write(t1curve,file="ADP_startValue.txt",ncolumns=2)

startLastVcount <- .95*maxN
lastV <- v[,startLastVcount:maxN]
means <- array(0, Deadline + 1)
for (step in 1:Deadline) means[step] <- mean(lastV[step,])
plot(means)
plot(main="ExpSmoothed SSE (alpha=0.01), t=1",ssqe[1,], xlab = "Iteration", ylab = "SSE",type = "l")
plot(main="thetaOne t=1",theta[1,])
plot(main="thetaTwo t=1",thetaTwo[1,])
plot(main="v = 1",v[1,])
startValue[cap/plotStep+1]
```

APPENDIX C. WPSP 2SSIP SAMPLE AVERAGE APPROXIMATION CODE

As described in the main body of this dissertation, we employed the statistical programming language R for all the code in this dissertation. From the R code we called the Gurobi solver. The SIP code employs an adaptation of the Sample Average Approximation algorithm developed by Kleywegt, Shapiro, and Homem-De-Mello (2001) and as described in Chapter 5.

```
# This is the SAA algorithm
# allows for running multiple scenairo cases
# This file generates all the data needed
# It contains the model logic
# It makes the call to gurobi and generates the output
# All of this is embedded in a for loop so that we can solve multiple instances of same SIP

library("gurobi")
library("Matrix")
setwd("~/Documents/GMU/Dissertation/Computational Approaches/StochIP/R Scripts")
start.time <- proc.time()  #initializes timing of the loop to give us a run time measure.
last.time <- proc.time() #timer log check

reDo <- TRUE

set.seed(301) #seed for default = ?
#basic parameters
deadline <- 12#50
capacity <- 100#2500000000#100
scaler <- 1#000000
capacityScaled <- capacity/scaler
Nset <- c(250)#,500,1000,2500,5000,10000,25000)#,10000)
initiatives <- 1
```

```
#data collection
reps <- length(Nset)
saaSample <- 10 # must be greater than 1
scenarios <- max(Nset) #how many scenarios per instance
epsilonAverage <- rep(0,reps)
defaultStop <- 20
worstCaseRun <- deadline*scenarios*saaSample*defaultStop
zeros <- rep(0,deadline)
initiativeZeros <- rep(0,initiatives)

weekIndex <- 1:deadline
initiativeIndex <-1:initiatives
futureIndex <-1:scenarios
epsilon <- capacityScaled/1000
cost<-array(0,initiativeIndex)
value<-array(0,initiativeIndex)
discountFactor<-array(1,deadline)
runTimeLog <- array(0,c(reps+1,5))
outputTable <- array(0,c(reps*saaSample*defaultStop,6))
saaStopTable <- array(0,c(reps*saaSample,7))
meanSaaStopTable <- array(0,c(reps,5))

#Generate future arrivals data
p <- 1/3
meanLogNorm <-2#16.58
varLogNorm <- 0.5#3.57
sdLogNorm <- varLogNorm^.5
meanArrival <- exp(meanLogNorm + .5*varLogNorm)
rate <- 1/meanArrival

if (reDo) {
  worstCaseCost <-
round(rlnorm(worstCaseRun,meanLogNorm,sdLogNorm)*(runif(worstCaseRun)<p)/scaler,digits = 1)
  worstCaseValue <- round(worstCaseCost*2*runif(worstCaseRun),digits=1)
}

# interim.time <-proc.time() - last.time
last.time <- proc.time()
# runTimeLog[1,1] <- "prep"
# runTimeLog[1,2] <- interim.time[3]

for (rep in 1:reps) {
  scenarios <- Nset[rep]
  sampleSize <- deadline*scenarios
  saaSampleSize <- deadline*scenarios*defaultStop
  nCols = initiatives + sampleSize
  nRows = 1 + scenarios
  meanStopCount <- 0
```

```
  meanStopTime <- 0
  mumu <- 0
  K <- p*capacity
  summu <- 0
  sumSqr <- 0

  for (iteration in 1:saaSample) {
    for (trial in 1:defaultStop) {
      step <- trial + defaultStop*(iteration-1) + saaSample*defaultStop*(rep-1)
      trialRun <- 1:sampleSize + sampleSize*(trial-1) + saaSampleSize*(iteration-1)
#       print(trialRun)
      futCost <- worstCaseCost[trialRun]
      futValue <- worstCaseValue[trialRun]

#       preMatrix.time <- proc.time() - last.time
#       last.time <- proc.time()

      #create the sparse matrix
      rowNonZeros <- initiatives+deadline
      totalNonZeros <- rowNonZeros*scenarios
      i <- c(rep(1,initiatives),ceiling((1:totalNonZeros)/rowNonZeros)+1)
      j <- rep(0, initiatives + totalNonZeros)
      x <- rep(0, initiatives + totalNonZeros)
      j[initiativeIndex] <- initiativeIndex
      x[initiativeIndex] <- cost
      for (k in 1:scenarios) {
        startPt <- (k-1)*rowNonZeros + 1 + initiatives
        endPt <- startPt + initiatives - 1
        j[startPt:endPt] = initiativeIndex
        x[startPt:endPt] = cost
        startPt <- endPt + 1
        endPt <- startPt + deadline - 1
        thisScenarioIndex <- 1:deadline + (k-1)*deadline
        j[startPt:endPt] <- 1:deadline + initiatives + (k-1)*deadline
        x[startPt:endPt] <- futCost[thisScenarioIndex]
      }

      Amatrix <- sparseMatrix(i,j,x=x)

#       matrix.time <- proc.time() - last.time
#       last.time <- proc.time()

      # create the objective values vector
      Vvector <- c(value,(1/scenarios)*futValue)

      # build rhs
      RHS <- rep(capacityScaled,scenarios+1)
      inEquality <- rep("<=",scenarios+1)
```

```
    model <- list()

    model$A          <- Amatrix
    model$obj        <- Vvector
    model$modelsense <- "max"
    model$rhs        <- RHS
    model$sense      <- inEquality
    model$vtype      <- 'B'

    params <- list(OutputFlag=0, MIPGap = 0.0001)

#     build.time <-proc.time() - last.time
#     last.time <- proc.time()

    result <- gurobi(model, params)
    sipSolution <- result$objval


    #   solFirst <- result$x[1:initiatives]
    #   solSecond <- result$x[(initiatives+1):length(result$x)]
    #   solMatrix <- matrix(solSecond,nrow = scenarios, ncol = deadline, byrow = T)
    #   print(solMatrix)
    #   print(futCost)

    # update estimate of mu

    if (trial == 1) {
      mu <- sipSolution
      muLast <- capacity
    } else {
      muLast <- mu
      mu <- sipSolution*(1/trial) + ((trial-1)/trial)*muLast
    }

    # update estimate of error
    error <- abs(mu-muLast)


    # update outputTable
    outputTable[step,1] <- scenarios
    outputTable[step,2] <- iteration
    outputTable[step,3] <- trial
    outputTable[step,4] <- sipSolution
    outputTable[step,5] <- mu
    outputTable[step,6] <- error

    # check SAA stopping condition - stop for loop if met
```

```
    if (error < epsilon) {
      step <- iteration + (rep-1)*saaSample
      runTime <-proc.time() - last.time
      last.time <- proc.time()

      saaStopTable[step,1] <- rep
      saaStopTable[step,2] <- scenarios
      saaStopTable[step,3] <- iteration
      saaStopTable[step,4] <- trial
      saaStopTable[step,5] <- runTime[3]
      saaStopTable[step,6] <- mu
      saaStopTable[step,7] <- error



      meanStopCount <- (iteration-1)/iteration*meanStopCount + 1/iteration*trial
      meanStopTime <- (iteration-1)/iteration*meanStopTime + 1/iteration*runTime[3]
      mumu <- (iteration-1)/iteration*mumu + 1/iteration*mu
      shiftMu <- mu - K
      summu <- summu + shiftMu
      sumSqr <- sumSqr + shiftMu*shiftMu

      break
    }
#      runTimeLog[rep+1,1] <- rep
#      runTimeLog[rep+1,2] <- preMatrix.time[3]
#      runTimeLog[rep+1,3] <- matrix.time[3]
#      runTimeLog[rep+1,4] <- build.time[3]
#      runTimeLog[rep+1,5] <- runTime[3]
   }
  }
  meanSaaStopTable[rep,1] <- Nset[rep]
  meanSaaStopTable[rep,2] <- meanStopCount
  meanSaaStopTable[rep,3] <- meanStopTime
  meanSaaStopTable[rep,4] <- mumu
  meanSaaStopTable[rep,5] <- (sumSqr - summu*summu/saaSample)/(saaSample-1)
}


end.start<- proc.time()-start.time
print(c("Runtime=",end.start))
# print(outputTable)
print(saaStopTable)
print(meanSaaStopTable)
```

REFERENCES

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* New Jersey: Prentice-Hall, Inc., 1993.

Artzner, P., F. Delbaen, J. M. Eber, and D. Heath. Thinking coherently. *Risk*, **10** (1997), 68-71.

Bellman, R. E. *Dynamic Programming*. Princeton: Princeton University Press, 1957.

Beale, E. L. M. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society Series B*, **17** (1955), 173–184.

Benders, J. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, **4** (1962), 238–252.

Bertha, R. L. and R. L. Shelton. "Combat Operations Analysis." In *Methods for Conducting Military Operational Analysis*, edited by A. G. Loerch and L. B. Rainey, Washington: Military Operations Research Society, 2007.

Birge, J. R. and F. Louveaux. *Introduction to Stochastic Programming*. New York: Springer-Velag, 1997.

Brown, G. G., R. F. Dell, A. G. Loerch, A. M. Newman. "Optimizing Capital Planning." In *Methods for Conducting Military Operational Analysis*, edited by A. G. Loerch and L. B. Rainey, Washington: Military Operations Research Society, 2007.

Charnes, A. and W. W. Cooper. Deterministic Equivalents for Optimization and Satisficing under Chance Constraints. *Operations Research*, **11** (1963), 18-39.

Chen, C. H. "An Effective Approach to Smartly Allocate Computing Budget for Discrete Event Simulation." Proceedings of the 34th IEEE Conference on Decision and Control, pp. 2598-2605, December, 1995.

Cohn, A. and C. Barnhart. The stochastic knapsack problem with random weights: A heuristic approach to robust transportation planning. *Proceedings of the Triennial Symposium on Transportation Analysis (TRISTAN III)*, (1998).

Cord, J. A Method for Allocating Funds to Investment Projects when Returns Are Subject to Uncertainty. *Management Science*, **10** (1964), 335-341.

Crain, W. F. "Theater Campaign Analysis." In *Methods for Conducting Military Operational Analysis*, edited by Andrew G. Loerch and Larry B. Rainey, Washington: Military Operations Research Society, 2007.

Dantzig, G. B. Linear Programming under Uncertainty. *Management Science*, **1,** (1955), 197–206.

Dantzig, G. B. Discrete–Variable Extremum Problems. *Operations Research*, **5** (1957), 266–277.

Dean, B. C., M. X. Goemans, and J. Vondrak. Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity. *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science* (2004), 208-217.

Dell, R. F. and W. F. Tarantino. How Optimization Supports Army Base Closure and Realignment. Technical Report, NPS-OR-03-003-PR, Naval Postgraduate School, 2003.

Denardo, E. V. *Dynamic Programming: Models and Applications*. Mineola, NY: Dover Publications, Inc., 2003.

Dinkelbach, W. On nonlinear fractional programming. *Management Science*, **13** (1967), 492-498.

Ellis, R. F., R. D. Rogers, and B. M. Cochran. Joint Improvised Explosive Device Defeat Organization (JIEDDO): Tactical Successes Mired in Organizational Chaos; Roadblock in the Counter-IED Fight. Thesis, Joint Forces Staff College, National Defense University. 2007.

Fortz, B., M. Labbé, F. Louveaux, and M. Poss. A non-linear approach to the stochastic knapsack problem with recourse. *VI ALIO/EURO Workshop on Applied Combinatorial Optimization*, (2008).

Freeman, P. R. The secretary problem and its extensions: A review. *International Statistical Review / Revue Internationale de Statistique*, **51** (1983), 189-206.

Freund, R. M. "Benders' Decomposition Methods for Structured Optimization, including Stochastic Optimization." Lecture Notes, Massachusetts Institute of Technology, 2004.

Greenberg, H. J. Dynamic Programming with Linear Uncertainty. *Operations Research,* **16** (1968), 675-678.

Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual.* 2015. http://www.gurobi.com

Henig, M. Risk Criteria in a Stochastic Knapsack Problem. *Operations Research*, **38** (1990), 820-825.

Howard, R. A. *Dynamic Programming and Markov Processes*. New York: Wiley, 1960.

Joint Improvised Explosive Device Defeat Organization. *Joint Improvised Explosive Device Defeat (JIEDD) Capability Approval and Acquisition Management Process (JCAAMP)*. Washington: JIEDDO Instruction 5000.01, 2007.

Joint Improvised Explosive Device Defeat Organization. *Annual Report FY 2008*. Washington: 2009.

Joint Improvised Explosive Device Defeat Organization. *JIEDDO Strategy for FY09-10*. Washington: 2009.

Joint Improvised Explosive Device Defeat Organization. *Annual Report FY 2010.* Washington, 2011.

Kaplan, E. H. and M. Kress. Operational Effectiveness of Suicide-Bomber-Detector Schemes: A Best-Case Analysis. *Proceedings of the National Academy of Sciences of the United States of America*, **102** (2005),10399-10404.

Keeney, R. L. and H. Raiffa, H. *Decision Making with Multiple Objectives: Preferences and Value Tradeoffs.* New York: Wiley, 1976.

Keles, P. and J. C. Hartman. Evaluating Portfolios of Multi-Stage R&D Portfolios with Approximate Dynamic Programming. Industrial and Systems Engineering Technical Report No. 07T-002, Lehigh University, 2007.

Kirkwood, C. W. *Strategic Decision Making: Multiobjective Decision Analysis with Spreadsheets*. Belmont, CA: Duxbury Press, 1997.

Kleywegt, A. J. Dynamic and Stochastic Models with Freight Distribution Applications. Ph.D. thesis, School of Industrial Engineering, Purdue University, 1996.

Kleywegt, A. J. and J. D. Papastavrou. The Dynamic and Stochastic Knapsack Problem. *Operations Research*, **46** (1998), 17-35.

Kleywegt, A. J. and J. D. Papastavrou. The Dynamic and Stochastic Knapsack Problem with random sized items. *Operations Research*, **46** (2001), 17-35.

Kleywegt, A. J., A. Shapiro, T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, **12** (2001), 479–502.

Kolesar, P. J. A Branch and Bound Algorithm for the Knapsack Problem. *Management Science*, **13** (1967), 723-735.

Konedaris, G. Value Function Approximation in Reinforcement Learning using the Fourier Basis. (2008). University of Massachusetts – Amherst, Computer Science Department Faculty Publication Series. Paper 101. http://scholarworks.umass.edu/cs_faculty_pubs/101.

Lewis, T. P., D. A. Fulk, and G. Castro. "Analysis of Alternatives." In *Methods for Conducting Military Operational Analysis*, edited by A. G. Loerch and L. B. Rainey, Washington: Military Operations Research Society, 2007.

Loerch, A. G., R. R. Koury, and D. T. Maxwell. Value Added Analysis for Army Equipment Modernization. *Naval Research Logistics*, **46** (1999), 233-253.

Lu, L. L., S. Y. Chiu, and L. A. Cox, Jr. Optimal project selection: Stochastic knapsack with finite time horizon. *Journal of the Operational Research Society*, **50** (1999), 645-650.

Markowitz, H. M. Portfolio Selection. *The Journal of Finance*, **7** (1952), 77–91.

Markowitz, H. M. *Portfolio Selection: Efficient Diversification of Investments*, New York: John Wiley. 1959.

McCabe, B. Monte Carlo Simulation for Schedule Risks, *Proceedings of the 2003 Winter Simulation Conference* (S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, Editors), Institute of Electrical and Electronics Engineers, New Orleans, Louisiana, USA, 7–10 December, 1561-1565.

Morison, S. E. *The Two-Ocean War*, New York: Little Brown and Co. 1963.

Morita, H., H. Ishii, and T. Nishida. Stochastic linear knapsack programming problem and its application to a portfolio selection problem. *European Journal of Operational Research*, **40** (1989), 329-336.

Morton, D. P. and R. K. Wood. "On a Stochastic Knapsack Problem and Generalizations." In *Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search*, edited by D. L. Woodruff, New York: Springer, 1998.

Nash, S. G. and A. Sofer. *Linear and Nonlinear Programming*, Singapore: McGraw-Hill. 1996.

Orgeron, H. J. "Analysis of Smaller Scale Contingencies." In *Methods for Conducting Military Operational Analysis*, edited by A. G. Loerch and L. B. Rainey, Washington: Military Operations Research Society, 2007.

Papastavrou, J.D., S. Rajagopalan, and A. J. Kleywegt. The Dynamic and Stochastic Knapsack Problem with Deadlines. *Management Science*, **42** (1996), 1706-1718.

Parnell, G. "Value-Focused Thinking." In *Methods for Conducting Military Operational Analysis*, edited by A. G. Loerch and L. B. Rainey, Washington: Military Operations Research Society, 2007.

Parnell, G.S., G.E. Bennett, J.A. Engelbrecht, R. Szafranski. Improving resource allocation within the National Reconnaissance Office. *Interfaces*, **32** (2002), 77-90.

Parnell, G. S. et al. Air Force Research Laboratory Space Technology Value Model: Creating Capabilities for Future Customers. *Military Operations Research*, **9** (2003), 5-17.

Perry, W. "Linking Systems Performance and Operational Effectiveness." In *Methods for Conducting Military Operational Analysis*, edited by A. G. Loerch and L. B. Rainey, Washington: Military Operations Research Society, 2007.

Powell, W. B. *Approximate Dynamic Programming (2nd ed)*, Hoboken: John Wiley & Sons, 2010.

R Core Team. *R: A Language and Environment for Statistical Computing,* Vienna: R Foundation for Statistical Computing, 2014. http://www.R-project.org

Ross, K.W. and D.H.K. Tsang. The stochastic knapsack problem. *IEEE Transactions on Communications*, **37** (1989), 740-747.

Ross, S. M. *Introduction to Probability Models (6th ed)*, San Diego: Academic Press, 1997.

Salkin, H. M. and C. A. de Kluyver. The knapsack problem: a survey. *Naval Research Logistics Quarterly*, **22** (1975), 127-144.

Turnbull, D. "Portfolio Management Project Plan." Presentation to JIEDDO J-9, Arlington, VA, 30 October, 2008.

U.S. Congress. House of Representatives. Committee on Armed Services. *Joint Improvised Explosive Device Defeat Organization: DoD's Fight Against IEDS Today and Tomorrow.* Committee Print 110-11, Nov 2008.

U.S. Government Accountability Office. *Spending Patterns of the Departments and Agencies of the Federal Government.* PAD 80-34. Washington: Government Printing Office, Dec 1979.

U.S. Government Accountability Office. Report to Congressional Committees. *Defense Management: More Transparency Needed Over the Financial and Human Capital Operations of the Joint Improvised Explosive Device Defeat Organization.* Washington: Government Printing Office, Mar 2008.

U.S. Government Accountability Office. Report to Congressional Committees. *Challenges Confronting DOD's Ability to Coordinate and Oversee Its Counter-Improvised Explosive Device Efforts.* Washington: Government Printing Office, Oct 2009.

Van Slyke, R., and Young, Y. Finite Horizon Stochastic Knapsacks with Applications to Yield Management. *Operations Research*, **48** (2000), 155-172.

Washburn, A. R. *Search and Detection*, *(4th ed)*. Linthicum: Institute for Operations Research and the Management Sciences, 2002.

Washburn, A.R. Continuous Network Interdiction. Technical Report, NPS-OR-06-007, Naval Postgraduate School, 2006.

Welford, B. P. Note on a method for calculating corrected sums of squares and products. *Technometrics*, **4**(3) (1962), 419–420.

Wilbaut, C., S. Hanafi, and S. Salhi. A survey of effective heuristics and their application to a variety of knapsack problems. *IMA Journal of Management Mathematics*, **19** (2008), 227−244.

BIOGRAPHY

Ronald F. A. "Fred" Woodaman was born at Portsmouth Naval Hospital, Norfolk , VA. His father, Ronald E. H. Woodaman, USNA '59, was a Navy frogman, then a professor of linguistics, and finally a career intelligence officer. His grandfather, RADM Ronald J. Woodaman, USN (Ret.), USNA '31, commanded two destroyers in World War II, fighting in both the Atlantic and the Pacific. His mother, Nivia Luz Woodaman nee Torres de Estanga, is a proud naturalized American citizen, having been born and raised in Venezuela. Fred spent his childhood in Maracay, Venezuela, returning to the United States to graduate high school from J.W. Robinson, Jr. Secondary School in Fairfax, VA. He graduated from the U.S. Naval Academy with Bachelor of Science in Systems Engineering in 1987. He served 20 years in the Marine Corps as an infantry officer, participating in operations in Panama, DESERT STORM, and IRAQI FREEDOM. Mid-career, he attended the Naval Postgraduate School, Monterey, CA, earning a Master of Science (with Distinction) in Operations Analysis. He retired from the Marine Corps in 2007, eventually becoming a Research Assistant from 2007-2010, with George Mason University's C4I Center. There he worked chiefly on a research contract for the Joint Improvised Explosive Device Defeat Organization, which provided the genesis for his dissertation. Subsequently he worked for Innovative Decisions Incorporated, where he led a variety of operations research projects for the Navy and Marine Corps until late 2013. He currently works for NTVI Federal Inc., providing operations research support to the Defense Suicide Prevention Office. He is married to the former Michelle L. Wolpert, with whom he has three children: Zoe, Iain, and Isabella. They reside in Stafford, VA, where he is active in youth sports and church activities. Taught by his father, he remains a life-long devotee of the Great Highland Bagpipe.