Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Optimal Decisions for Canadian Military Airlift Problem

Abdeslem Boukhtouta, Jean Berger
(DRDC Valcartier)

Warren B. Powell, Belgacem Bouziane-Ayari
(Princeton University)

Defence Research and Development Canada – Valcartier

Technical Report
DRDC Valcartier TR 2010-517
September 2010

Canada

Optimal Decisions for Canadian Military Airlift Problem

Abdeslem Boukhtouta, Jean Berger
(DRDC Valcartier)

Warren B. Powell, Belgacem Bouziane-Ayari
(Princeton University)

Defence Research and Development Canada – Valcartier

Technical Report
DRDC Valcartier TR 2010-517
September 2010

IMPORTANT INFORMATIVE STATEMENTS

Abstract

A modeling and algorithmic framework for analyzing and simulating the military airlift is presented in this report. Central to the framework is the modeling of the flow of information and decisions. A series of models demonstrating how different levels of information can be represented when making a decision are presented. These information sets are simulated to demonstrate their impact on costs and throughput. A detailed comparison of classical optimization and the current framework is presented. Using historical data for the airlifts conducted by the Canadian Air Force, a series of simulations were conducted to test the effect of uncertainty in customer demands as well as aircraft failures. It is demonstrated in this report that this effect is reduced when adaptive learning is used in the simulation process. There are a number of sources of randomness that arise in military airlift operations: random demands, aircraft failures, weather delays, etc. The cost of uncertainty can be difficult to estimate because it is difficult to give the mathematical expression of this cost. However, it is easy to overestimate this cost if we use simplistic decision rules. The military airlift problem, considered in this report, is modeled as a dynamic program, and solved using approximate dynamic programming. The experiments show that even approximate solutions produce decisions that substantially reduce the effect of uncertainty.

Résumé

Une approche de modélisation et de résolution pour l'analyse et la simulation du problème du transport aérien militaire est présentée dans ce rapport. La modélisation des flots d'informations et de décisions en constitue l'élément central. Nous présentons plusieurs modèles pour montrer comment l'information est traitée et présentée durant le processus de prise de décision. Plusieurs scénarios sont simulés pour déterminer l'impact de l'information sur les coûts ainsi que sur la solution. Nous faisons une comparaison détaillée entre l'approche d'optimisation classique et celle qui est proposée dans ce rapport qui est le *simulateur d'optimisation*. En utilisant des données historiques des opérations aériennes des Forces Canadiennes, une série de simulations a été effectuées pour mesurer l'impact des incertitudes liées aux requêtes de dernière minute et aux bris mécaniques des avions. Nous démontrons que cet impact est réduit lorsque nous utilisons le concept d'apprentissage adaptatif dans le processus de simulation. Il y a plusieurs sources d'incertitude liées au problème du transport militaire aérien : requêtes imprévues, bris d'avions, retards dus à la météo, etc. L'estimation du coût d'incertitude est difficile à obtenir puisque ce dernier est difficile à exprimer mathématiquement. Par ailleurs, ce coût est facile à surestimer quand on utilise des règles de décisions simplistes. Le problème du transport militaire aérien abordé dans ce rapport est modélisé comme un programme dynamique qui est résolu via la programmation dynamique approximative. Les expériences numériques montrent que même les solutions approximatives donnent des décisions qui réduisent substantiellement l'effet d'incertitude.

This page intentionally left blank.

Executive summary

Optimal Decisions for Canadian Military Airlift Problem

A. Boukhtouta; J. Berger; W. B. Powell; B. Bouzaiene-Ayari; DRDC Valcartier TR 2010-517; Defence R&D Canada – Valcartier; September 2010.

The Canadian military airlift problem involves managing a fleet of aircraft to serve demands to move passengers or freight. The aircraft are characterized by attributes, some of which may be dynamic, such as the current location, the earliest time when it can be used to serve a demand, whether it is currently loaded or empty, and so on. A demand is also described using features such as type, origin, aircraft type required to serve the demand, priority, and arrival and departure times. Aircrafts are used to execute decisions such as pick up and move missions, and can be reallocated to alternate locations to serve emerging demands. Demand is not fully deterministic, meaning that last minute requests may occur in some particular locations. Other forms of uncertainty, such as random equipment failures, may also happen.

The goal of this study is to compare, for the military airlift problem, the performance and capabilities of the optimizing-simulator technology (handling uncertainties) against that of classical deterministic optimization. The optimizing-simulator, rather than fostering a competition between alternative analysis technologies, actually combines the best features of simulation and optimization. In the process, it combines cost-based and rule-based decision technologies, providing an analysis technology that handles the high-dimensionality of problems that typically arise in transportation, as well as the complex behaviours that may be difficult to capture using a cost model, but which can be expressed as patterns of behaviour.

The optimizing-simulator allows us to run studies on ‘‘what if’’ questions such as:

- What is the effect of random customers’ demands over time?
- What is the value of knowing all orders in advance?
- What is the effect of equipment failures, delays due to weather or problems at airbases?

The report provides a tutorial on how to model the flow of information and the airlift complex problem dynamics using a simple, intuitive vocabulary. We then demonstrate this modeling framework in the context of the Canadian military airlift problem. Finally, we present a series of computational experiments that evaluate the effect of random demands and equipment failures on system performance. The experiments clearly demonstrate the capability of the optimizing-simulator. In terms of validating the methodology, the main outcome is that computational results are all intuitively reasonable. We observed that increasing the level of uncertainty (random demands, aircraft breakdowns or both) reduces solution quality. Also, the adaptive learning approach is found to improve solution quality as well as to reduce the cost of uncertainty.

Sommaire

Optimal Decisions for Canadian Military Airlift Problem

A. Boukhtouta ; J. Berger ; W. B. Powell ; B. Bouzaiene-Ayari ; DRDC Valcartier TR 2010-517, R & D pour la défense Canada – Valcartier ; Septembre 2010.

Le problème de transport aérien des Forces canadiennes consiste à gérer une flotte d'avions pour répondre à des requêtes (missions) de transport de passagers et de marchandises. Les avions sont caractérisés par des attributs (ex. localisation, disponibilité prochaine pour une mission, état de chargement) qui peuvent varier en fonction du temps. Une requête ou une mission peut aussi être décrite en fonction de certaines caractéristiques telles que type de demande, origine, type d'avion requis pour accomplir cette mission, priorité, périodes d'arrivée et de départ. Les avions sont utilisés pour exécuter des décisions et missions du genre cueillette de marchandise et transport, et peuvent être réaffectés à un site alternatif pour répondre à une autre requête. Les requêtes associées aux différentes missions ne sont pas complètement déterministes indiquant que des requêtes de dernière minute peuvent survenir à des endroits quelconques. D'autres formes d'incertitudes, peuvent également se manifester, tels les bris d'équipement inattendus.

L'objectif de la présente étude est de comparer pour le problème militaire de transport aérien les performances d'un simulateur-optimiseur (considérant les incertitudes) avec l'optimisation déterministe classique (sans incertitude). Le simulateur-optimiseur combine les avantages de l'optimisation et de la simulation dans la résolution des problèmes. Il combine, dans le processus de résolution, une approche basée sur les coûts avec une autre approche basée sur les règles de décision, pour donner naissance à un outil d'analyse qui peut traiter les problèmes de grandes tailles comme ceux du transport. Aussi, il permet de tenir compte des caractéristiques complexes qui peuvent être difficiles à modéliser en utilisant une approche basée sur les coûts.

Le simulateur-optimiseur permet d'accomplir des analyses par simulation/anticipation telles que :

- Quel est l'effet de réception aléatoire des requêtes?
- Quelle est la valeur de la connaissance a priori de l'information relative aux requêtes?
- Quel est l'effet de bris des équipements, des retards dus au climat ou aux problèmes relatifs aux bases aériennes?

Le présent rapport présente un cadre de travail traitant de la dynamique de modélisation des flots d'information et du problème de transport aérien en utilisant un vocabulaire simple et intuitif. Nous appliquons ce cadre de travail à la modélisation et la résolution du problème de transport militaire aérien. Nous présentons une série d'expériences numériques pour évaluer l'effet de la réception aléatoire des requêtes et des bris des équipements sur la performance du système. Les expériences numériques démontrent bien les capacités du simulateur-optimiseur. En matière de validation de la méthodologie, les résultats obtenus sont logiques et raisonnables. Nous avons remarqué que l'augmentation du niveau d'incertitude (requêtes aléatoires, bris mécanique de l'avion ou les deux) réduit la qualité de la solution. Aussi, les solutions données par l'approche basée sur l'apprentissage adaptatif améliorent la qualité de la solution tout en réduisant le coût d'incertitude.

Table of contents

List of figures

List of tables

This page intentionally left blank.

1 Introduction

The Canadian military airlift problem involves managing a fleet of aircraft to serve demands to move passengers or freight. The aircraft are characterized by attributes, some of which may be dynamic, such as the current location, the earliest time when it can be used to serve a demand, whether it is currently loaded or empty and so on. The demands are also described using features such as the type of demand, their origin, and the type of aircraft that is required to serve the demand, their priority, and arrival and departure times. Aircrafts are used to execute decisions such as pick up and move missions, or reallocated to an alternate location to serve emerging demands. Demand is not fully deterministic, last minute requests may occur for any specific location. Other forms of uncertainty, such as random equipment failure may emerge as well.

The optimizing-simulator is an analysis technology that focuses on the modeling of complex resource allocation problems, using an explicit model of the information available to make a decision. We begin in this report with a brief presentation of how we model complex resource allocation problems, covering both the representation of the resources themselves and modeling the information.

We present in this report a series of models that illustrate how we model the information content of a decision (such as move aircraft, load aircraft, delay a demand, etc.) and the value of increasing this information content. The results show that providing more information reduces costs and increases throughput. Also reported are the results of research comparing the objective function produced by the optimizing-simulator to optimal solutions for deterministic problems, and the results of rolling-horizon experiments for problems which exhibit uncertainty.

An important feature of the technology used in this report, is its ability to combine cost-based and rule-based control technologies. Rules are expressed in the form of low-dimensional patterns ("try to send C-17s to Europe"), and we penalize deviations from these patterns.

A detailed comparison of classical optimization approach and the optimizing-simulator is presented, focusing on both solution quality and the ability of the optimizing-simulator to model dynamic information processes, complex operational dynamics, and the information content of a decision. We then show how this modeling framework can be used to represent the Canadian military airlift problem. We should mention that the dataset used to solve the problem have been prepared for an optimization model, and as such lacked the richness that we typically find in real applications.

Finally, a series of experiments were run to test the capabilities of the modeling and algorithmic framework. We start by demonstrating the quality of the solution achieved using the approximate dynamic programming algorithm. With no uncertainty, and with a modest level of advance information, a simple simulation provides very high quality solutions. We then assumed that generic demands became known over the simulation, varying the time interval between a new demand and the time it has to be served (the prebook time). We were able to quantify the value of knowing orders in advance (we assumed 0, 2 and 6 hours), and we also showed that the value of advance information is reduced if we use adaptive learning logic.

Approximate dynamic programming helps the model produce decisions that are robust, which is to say that they work well under different potential outcomes in the future. It might send five aircraft to handle work that could be covered by three or four aircraft in anticipation of possible failures. It might also position extra aircraft at locations which can handle potential failures in nearby locations. Our interest is in estimating the cost of different forms of uncertainty, and determining the extent to which robust decision making affects the cost of uncertainty. Our analysis focuses on two forms of uncertainty: uncertainty in customer requests, and the possibility of equipment failures that introduce additional delays in the movement of aircraft. This document analyzes the effect of these two forms of uncertainty (customer requests and equipment failures) and quantifies the effect of these forms of uncertainty using two simulation methods. The first method uses a standard simulation in which decisions are made myopically. The second uses approximate dynamic programming to produce more robust decisions. We then compare and contrast the cost estimates of uncertainty using both policies.

This report is organized as follows. Chapter 2 provides a model for the airlift mobility problem using the notation and vocabulary of dynamic resource management. In Chapter 3, we review a range of decision making strategies (implemented in the optimizing simulator), starting with elementary rule-based policies that are commonly used in simulation, and extending through myopic, cost-based policies, closing with the policies derived from solving dynamic programs. Classical optimization is compared to the proposed optimizing simulator approach as well. Chapter 4 reports on a series of simulations which address the question of estimating the cost of uncertainty, and how this estimate depends on the type of decision function used. The model related to the Canadian military airlift problem is presented in Chapter 5 and the results related to this model are presented in Chapter 6. Finally, some concluding remarks are given in Chapter 7.

2 Modeling airlift mobility problem

Different modeling strategies have been used to solve the military mobility problems (air and sea): deterministic linear programming, simulation, and stochastic programming. Ferguson & Dantzig are one of the first to apply mathematical models to air-based transportation [1]. Subsequently, numerous studies were conducted on the application of optimization modeling to the military airlift problem (see [2] for more details).

Military airlift (and sealift) problems have typically been modeled either as deterministic linear (or integer) programs or with simulation models which can handle various forms of uncertainty. Much of the work on deterministic optimization models started at the Naval Postgraduate School with the Mobility Optimization Model (See [3])). The development of THRUPUT which is a general airlift model but which does not capture time is described in [4]. THRUPUT and the Mobility Optimization Model were combined to produce THRUPUT II (See [5]) to obtain a model which accurately captured airlift operations in the context of a time-dependent model. A similar model was produced by RAND called CONOP (CONcept of OPerations), which is described in [6]. Both THRUPUT II and CONOP possessed desirable features which were merged to a system called NRMO (See [7]). These models could all be solved using linear programming packages. The problem of routing and scheduling vehicles in the context of theatre of operations is addressed in [8]. The resulting model was solved using group-theoretic Tabu Search.

Despite the attention given to mathematical programming-based models, there remains considerable interest in the use of simulation models, primarily because of their ability to handle uncertainty as well as the flexibility in capturing complex operational issues. The Argonne National Laboratory developed TRANSCAP to simulate the deployment of forces from Army bases (Burke et al. [9]). TRANSCAP uses a discrete-event simulation module developed using the simulation language MODSIM III. The Air Mobility Command at Scott Air Force Base uses a simulation model, AMOS (derived from an earlier model known as MASS), to model airlift operations for policy studies.

As early as 1956 there has been interest in solving these problems where decisions explicitly account for uncertainty in the future [1]. Much of this work has evolved within the discipline of stochastic programming, which focuses on representing uncertainty within linear programs. The approach developed in [10] accounts for uncertainty in demands, while the study presented in [11] proposes an extension of THRUPUT II to handle uncertainty in aircraft reliability. Niemi proposes a stochastic programming model which captures uncertainty in ground times [12]. A Stochastic Sealift Deployment Model (SSDM) is introduced in [13] to model sealift operations in the presence of possible attacks.

A different line of investigation uses dynamic programming to find decisions in the presence of uncertainty. Yost & Washburn uses the theory of partially observable Markov decision processes to find decisions in the presence of uncertainty [14]. The resulting model, however, suffers from the classic curse of dimensionality, restricting its use to very small problems.

The framework proposed in this report is based on the field of approximate dynamic programming (See [15], [16], [17] and [18]) which solves Bellman's equation by approximating the value function using statistical methods. This general strategy has been adapted to multistage, stochastic optimization problems in order to combine the power of linear programming (which is needed to handle the high dimensional problems that arise in transportation) with the power of dynamic programming (see, for example [19]). This algorithmic strategy has been demonstrated in the context of a variety of fleet management problems (See [20], [21], [22], [23] and [24]), including the military airlift problem (See [25] and [26]). In addition to the academic research literature, these techniques have also been used in industrial truckload trucking, rail (boxcars and locomotives) and business jets applications.

We describe below the most important dimensions facing a resource allocation problem such as airlift mobility problems. These include:

- The resources being managed.
- The information available when making a decision.
- The decisions that act on the resources being managed
- The transition function that describes how the system evolves over time.
- Performance measures.

This discussion is based on a general model for resource allocation problems called the *dynamic resource transformation problem* (DRTP). A DRTP is comprised of three major dimensions which are represented using:

Knowledge || Processes || Controls

A detailed summary of this problem class is given in [27]. This presentation captures only the most important elements of this problem. Our modeling approach is developed to handle all of the dimensions (and subdimensions) of a full-scale DRTP.

2.1 Resources

The airlift mobility problem is often modeled using two resource classes, or *layers*, which might be the aircraft and the requirements (the requests to move passengers or freight). A third resource layer might be the pilots. Other resource layers could be maintenance capabilities or loading/unloading equipment. The resource layers are represented using

$$\begin{aligned} C^R &= \text{Set of resource layers (pilots, aircraft, demands).} \\ &= (P, A, D) \end{aligned}$$

Note that a resource layer describes anything that constrains the system over time. Thus, requirements in the form of freight or passengers limit our ability to move aircraft loaded. At the same time, our ability to satisfy requirements is limited by aircraft, pilots (as well as other potential limiting factors).

A resource layer is described using:

a = Attributes of a resource.

A^c =The set of potential attributes a for resources in class $c \in \mathrm{C}^R$.

We capture the state of our resources using:

R_{ta}^c = Number of resources in class $c \in \mathrm{C}^R$ with attribute a that we *know about* at time t.

$R_t^c = \left(R_{ta}^c\right)_{a \in \mathrm{A}^c}$ = Resource state vector at time t for resources in class $c \in \mathrm{C}^R$.

$R_t = \left(R_t^c\right)_{c \in \mathrm{C}^R}$ = Complete resource state vector at time t.

Thus, R_t^P would be our resource vector for pilots, R_t^A would be our resource vector for aircraft, and R_t^D would represent our set of demands.

It is important to understand the power of the attribute vector a for describing resources. We start by illustrating potential attributes for an aircraft. These might be:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = \begin{pmatrix} \text{Current or future location} \\ \text{Estimated time of arrival} \\ \text{Aircraft type} \\ \text{Fuel level} \\ \text{Maintenance status} \\ \text{Load attributes} \\ \text{Pilot attributes} \end{pmatrix}$$

We can obtain *layered resources* when, for example, we put a pilot in an aircraft, or a load of freight in an aircraft. In our attribute model above, these would be captured by attributes a_6 and a_7. Thus, these attributes may actually be vectors of attributes themselves.

It is important to understand the "estimated time of arrival" field. It is very common to have a resource that will have a particular attribute in the future (e.g. it is flying toward a destination), where you cannot act on the resource until it has arrived at this point in time. This time is known (in our vocabulary) as the *actionable time*. The attribute vector may evolve over time (for example, it may be delayed due to weather, changing the actionable time, or there may be an equipment failure, changing the maintenance status. We would let:

a_t = The attribute of the resource with the information available at time t.

Here, we would represent time t as the *knowable time*. It is important to understand that throughout our presentation, time t represents what information is available, whereas the time at which something actually happens (such as, arriving at an airbase) is the actionable time.

The requirements to be satisfied (loads of freight, people to be moved) might be represented using:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} \text{Origin} \\ \text{Destination} \\ \text{Pickup time window} \\ \text{Delivery time window} \\ \text{Pounds} \\ \text{Cubic feet} \end{pmatrix}$$

Here we see the actionable time in a different format: the pickup time window (this might again be a vector, possibly consisting of earliest, latest and target pickup time). The earliest time (the beginning of the window) would be the actionable time. This means that the vector R_t^D represents the demands that we *know about* at time *t*. When the demands have to be served is part of the attribute vector. Sometimes (as we do in this document) it is useful to bring out the actionable time, at which point we would use notation such as

$R_{t,t'a'}$ = Number of resources that we know about at time t, that will have attribute a' at time t'.

$R_{tt'} = \left(R_{t,t'a'}\right)_{a' \in \mathrm{A}}$ Vector of resources that we know about at time t that are actionable at time t'.

Thus, we would say that $R_{tt'}$ is the vector of resources that we know about at time *t* that are actionable at time *t'*. The vector $R_{0t'}^D$ represents the demands that are known at the very beginning (the "strategic" demands) that have to be served at a time starting at *t'*, while $\left(R_{tt'}^D\right)_{t>0}$ is all the demands that become known during the simulation.

An important aspect of our mathematical model is the flexibility inherent in the use of the attribute vector. It is quite easy to add dimensions to the attribute vector. The software library is designed assuming that the attribute vector is sufficiently complex that we cannot enumerate the attribute space (even small problems can have an attribute space A with millions of elements; for complex resources such as pilots and aircraft the attribute space may be orders of magnitude larger).

2.2 The information process

There are two types of information processes that we consider. The first and most important is the arrival of customer requests to move freight and passengers. The second represents information such as weather delays and equipment failures that affect the evolution of the system over time (more on this later).

To begin, it is important to understand how we model time. We view information as arriving in continuous time, but for practical reasons, we model decisions as occurring in discrete time. The relationship between discrete and continuous time is given in Figure 1.

To describe the flow of customer requests for loads to be moved, let

$\hat{R}_{ta}^D$ = Number of customer demands with attribute a that first become known at time t.

$\hat{R}_t^D = \left(\hat{R}_{ta}^D\right)_{a \in \mathrm{A}^D}$

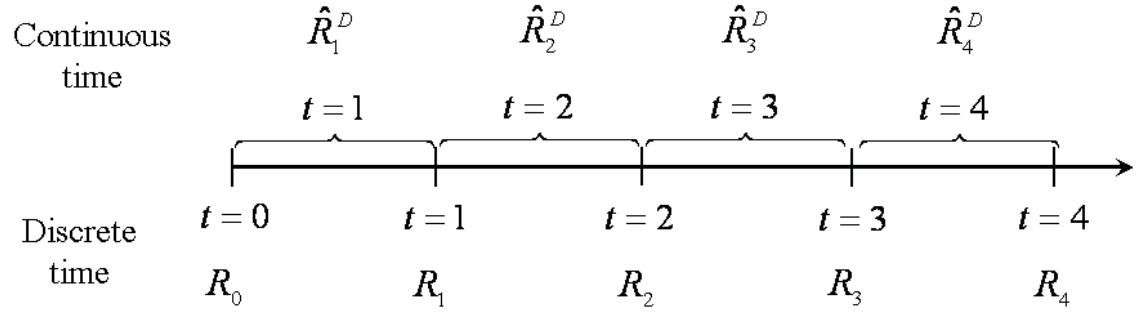

Figure 1: Relationship between continuous time information processes, and discrete time decision processes

$\hat{R}_t^D$ is the arrival process of customer requests over time. Referring to Figure 1, we see that $\hat{R}_t^D$ is defined only for t=1, 2, … We might also let

$\hat{R}_{tt'}^D$ = The vector of customer requests that arrive during time interval t that can first be served at time t'.

The difference between the call-in time t, and the actionable time t', is one of the major quantities we wish to address in this study. In addition to the arrival process for new demands, we may have other types of information that include weather delays and equipment failures. To keep our notation compact, we let

W_t = Family of random variables representing all possible types of new information (including customer demands) that may arrive during time interval t.

2.3 Decisions

For now, we represent decisions in a very simple way. We let:

C^D = Set of decision classes (move aircraft, load aircraft, repair aircraft, reconfigure an aircraft for passengers, delay a demand, rest a pilot)

D^c = Set of decision types in class $c \in \mathrm{C}^D$

D = Set of all decision types

$= \bigcup_{c \in \mathrm{C}^D} \mathrm{D}^c$

In transportation applications, the most common decision type is to move from one location to another. For this decision class, D represents the set of locations we can move to. But we can also handle decisions which change the configuration of an aircraft, refill an aircraft with fuel, or repair an aircraft.

The set of decisions typically depends on the attributes of the resource a decision is being applied to. For this reason, it is useful to define

$\mathrm{D}(a)$ = The decision set function, which returns a set of decisions given an attribute vector *a*.

Our decision vector is represented using

x_{tad} = Number of resources with attribute *a* that we act with a decision of type *d* using the information available at time *t*.

$x_t = (x_{tad})_{a \in \mathrm{A}, d \in \mathrm{D}}$

It is important to realize that we may make a decision *d* (for example, move from one location to another) acting on an aircraft (or a demand) that is not actionable at time *t*. For example, at time *t*, there may be an aircraft that is actionable now, which we wish to assign to a demand that is actionable at, say, *t*+10 (assume that it takes only 2 time periods to get from the current location of the aircraft to the origin of the demand). We assume that if you can implement the decision later and still arrive in time to handle the demand, that we delay implementing the decision until a later point in time.

In an optimizing-simulator, we are going to model a decision with variety of ways. For the moment, it is enough to say that we have a decision function that we represent using

$X_t^{\pi}(I_t)$ = Function returning a feasible vector x_t given information I_t. There may be a family of functions Π, where $\pi \in \Pi$ is one function.

When dealing with uncertainty, it is common to refer to a particular decision function as a *policy*. We often find ourselves comparing different rules for making decisions, which we refer to as comparing policies.

2.4 The transition function

In the vocabulary of dynamic systems, R_t represents the state of our system (to be precise, it is the *resource state* variable). For our purposes, our state variable is captured by the resource state vector. If W_t is the information arriving during time interval *t* and x_t is the vector of decisions, we can represent in a general way the evolution of our system over time using:

$$R_t = R^M(R_{t-1}, W_t, x_t)$$

In this representation, we are writing R_t as a *post-decision state variable*. That is, it is the state of the resources immediately after a decision is made but before new information is used. We

often use the post-decision state variable because it is easier to use algorithmically (see section 3.4). If we were to use a *pre-decision state variable*, we would write our recursion as:

$$R_{t+1} = R^M\left(R_t, x_t, W_{t+1}\right)$$

Sometimes, it is useful to use both pre- and post-decision state variables. In this case, we let R_t be the pre-decision state variable, and we let R_t^x be the post-decision state variable. We can break the transitions into two steps:

$$R_t^x = R^{M,x}\left(R_t, x_t\right)$$

$$R_{t+1} = R^{M,W}\left(R_t^x, W_{t+1}\right)$$

For most resource allocation problems, it is easier to work at the level of individual resources and decisions. When we act on a resource with attribute *a* with decision *d*, we model the results of this decision using:

$$M(a_{t-1}, W_t, d_t) \rightarrow (a_t, c, \tau)$$

$M(a_{t-1}, W_t, d_t)$ is known as the *modify function*, which captures how a resource is modified as a result of a decision. There are three outputs from this function: the modified attribute vector a_t, the cost or contribution of the decision (we use cost if we are minimizing, contribution if we are maximizing), and τ is the time required to complete the action. The terms a_t, c and τ are all computed using the information available at time t. $t+\tau$ would be the actionable time, which can be stored as an attribute, but sometimes it is helpful to have explicit notation for it.
The modify function is a simple, rule-based function which can handle virtually any level of complexity.

For mathematical modeling purposes, it is useful to define several terms that are based on the modified function. The first is the *terminal attribute function:*

$$a^M(a, W, d) = \text{The attribute vector } a' \text{ produced as a result of a decision } d \text{ acting on a resource with attribute } a \text{ with information } W.$$

The second is an indicator function for the terminal attribute function:

$$\delta_{a'}(a, W, d) = \begin{cases} 1 & \text{If } a^M(a, W, d) = a' \\ 0 & \text{Otherwise} \end{cases}$$

The function $\delta_{a'}(t, a, d)$ is useful for writing the equations that govern the evolution of the resource vector over time:

$$R_{ta'} = \sum_{a \in \mathrm{A}} \sum_{d \in \mathrm{D}} \delta_{a'}(a_{t-1}, W_t, d_t) x_{tad}$$

where flow conservation requires that

$$\sum_{d \in \mathrm{D}(a)} x_{tad} = R_{t-1,a} + \hat{R}_{t,a}$$

2.5 The objective function

In general, we can say that we have a cost function that captures all the costs associated with a vector of decisions. In practice, we typically find ourselves using a mixture of "hard dollar" costs (e.g. the cost of flying an aircraft to a particular location) and "soft dollar" costs (such as penalty for serving a demand late, or a bonus for using a particular type of aircraft to move passengers) which we add to the objective function to get the model to behave in a certain way. We represent the hard dollar cost function using

$$C(R_{t-1}, W_t, x_t) = \text{Total costs from entering time interval } t-1 \text{ with resource vector } R_{t-1}, \text{ combined with the new information } W_t \text{ and decision } x_t.$$

When we want to capture the presence of hard and soft dollar costs, we use

$$C^{\pi}(R_{t-1}, W_t, x_t) = \text{Total hard and soft dollar costs incurred during time interval } t.$$

We index the soft-dollar cost function by the same index π (π represents also a particular policy) since a set of soft dollar costs is typically associated with a particular policy. For example, we may put a bonus on serving a high priority demand. This bonus helps us decide whether to serve a lower priority demand that is closer (and therefore lower cost) or a higher priority demand that requires traveling longer distances. How this trade off is made constitutes a policy.

Our optimization problem is not one of choosing the best set of decisions, but rather choosing the best set of decision functions, or policies. Because of uncertainty, we cannot say with certainty what the state of the system will be in the future, so our problem is not to determine *what* we will do in the future, but rather *how* we will choose to make a decision. This problem is posed mathematically by writing

$$\min_{\pi \in \Pi} E \left\{ \sum_{t=1}^{T} C\left(R_{t-1}, W_t, X_t^{\pi}(R_{t-}, W_t)\right) \right\}$$

We illustrate in the next chapter a variety of decision functions.

3 The optimizing simulator

We now have the infrastructure we need to describe a variety of types of simulations that we can run with an optimizing simulator. These simulations will help to understand the relationship between classical simulation, classical optimization, and the optimizing-simulator framework (See [28]).

Our experiments progress in a series of steps which illustrate different decision technologies (rule-based and cost-based) and different levels of information available to make a decision. These technologies consist of the following (See [28]):

- Rule-based decision making – Here we consider rules that tell us "when in this state, take this action."
- A myopic cost-based policy – This is the first step toward using a decision function that requires a linear programming solver. It is restricted to optimizing over only what is known at a given point in time.
- Decisions using forecasts – This combines what we know right now with a forecast of future demands.
- Decisions with value functions – Here is where we first start using approximate dynamic programming. We capture the impact of decisions now on the future by using value function approximations.
- Decisions with patterns – Now we come full circle, combining cost-based logic with rule-based logic.

Each decision function requires its own information set. We illustrate the ideas through a series of examples that arise in the military airlift problem.

3.1 Rule-based decision making

Most classical simulation models use simple rules to make a decision: when in this state, take this action. In manufacturing simulations, a job might leave one machine, after which there might be a choice of several downstream machines where we might assign the job to the machine with the shortest queue.

In the airlift problem (as with many problems in transportation), it is typically necessary to juggle multiple aircraft at the same time. For example, Figure 2 shows four aircraft and four different demands that have to be served. It is impossible to design a rule that will handle all possible combinations of assigning different aircraft to different demands. Instead, we can simplify the problem by handling, say, one demand at a time. The "AMOS" simulator used by the Airlift Mobility Command at Scott Air Force Base uses logic that takes the demand that has to be moved first, then looks at the first available aircraft, and then determines whether it is feasible to assign the aircraft to serve the demand. If it is possible, then this is what happens. Another aircraft might be closer or better suited to the requirement. No attempt is made to find the "best" aircraft.

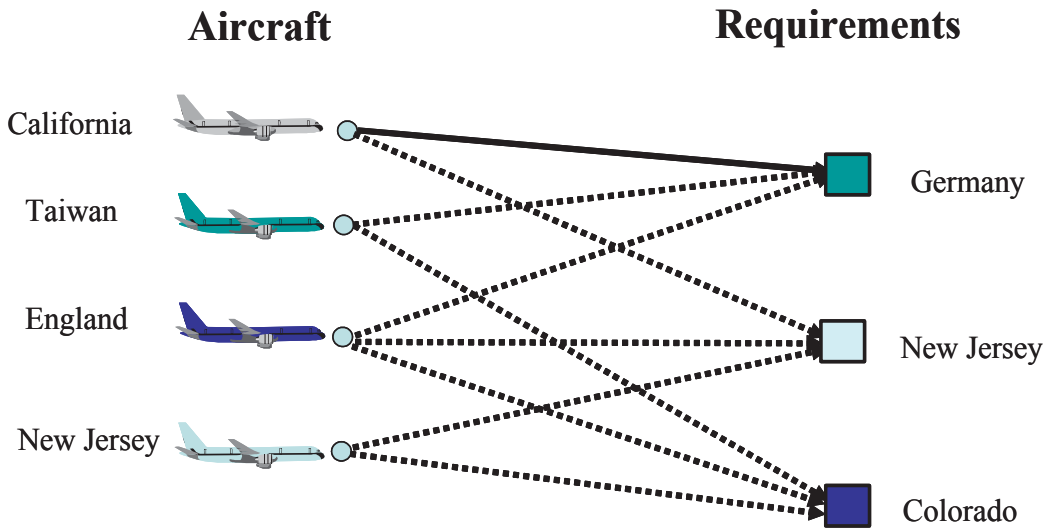

Figure 2: Multiple aircraft may be considered for multiple demands. Rule-based logic picks one possible assignment and determines feasibility (the solid line show that the Aircraft is assigned to the requirement)

If we apply our rule-based logic, we will produce (at a particular time *t*) one or more assignments of aircraft to demands. We represent these decisions using:

$$x_{tad} = \begin{cases} 1 & \text{If our rule specifies to use decision } d \text{ for an aircraft} \\ & \text{with attribute } a \\ 0 & \text{Otherwise} \end{cases}$$

Let I_t be the information that we had available to us when we made this decision at time *t* (in this case, the information is the set of available aircraft and demands), and let

$$X_t^{\pi}(I_t) = \text{A function returning a feasible vector } x_t.$$

Here, π represents a particular policy (in this case, the rules we described above), and we might let $\pi = \text{RB}$ to represent this rule-based policy.

3.2 A myopic cost-based policy

The next step is to introduce the use of a cost function. Figure 3 illustrates the different types of costs and bonuses that might be used to evaluate a decision to assign an aircraft to a demand. These will typically be a mixture of hard-dollar and soft-dollar costs. In practice, the transition from a rule-based to a cost-based system requires developing a cost model, which can be quite difficult in practice.

Issue	"cost"/"bonus"
Repositioning cost	-$17,000
Appropriate a/c type	+5000
Utilization	+8000
Requires modifications	-3000
Special maintenance at airbase	-1000
Total "cost"	-8000

Figure 3: Illustration of different elements of a cost of assigning an aircraft to a demand

If we can develop a cost function, we can perform a very simple optimization by considering a single demand and a list of aircraft that might be used to serve the demand. By ranking the aircraft based on cost, we can choose the best aircraft. The problem is illustrated in Figure 4.

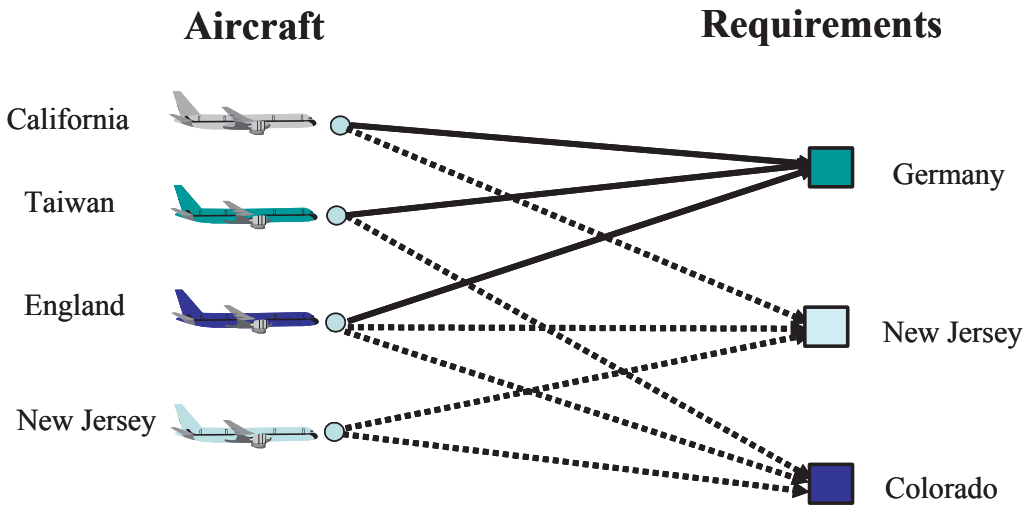

Figure 4: Choosing the best of a set of aircraft to serve a demand

Once we have made the transition to using a cost model to choose the best aircraft for a demand, we can consider multiple aircraft and demands (as illustrated in Figure 5) where we have to use a linear programming package to determine the optimal solution. These problems are typically quite small and can be solved very quickly. Most of the computer time arises in generating the network and computing the costs. For airlift problems, simply determining the cost of an option can be expensive.

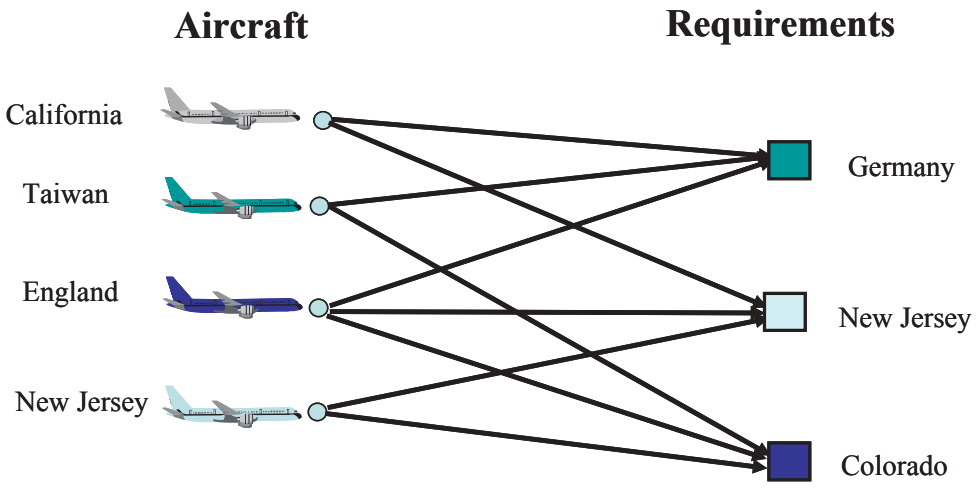

Figure 5: An assignment problem considering multiple aircraft and demands

Now that we have built up to a linear program, we need to return to the issue of ‘‘ knowability’’ versus ‘‘actionability’’. In our myopic problem, we assume that we are only considering the assignment of aircraft whose attributes are known to demands that are known. However, we have not discussed whether they are actionable now or later. We may know about an aircraft now, but it may be in the process of flying from one location to the next (or in the midst of a repair) and will not be actionable. Similarly, a customer demand may be known, but cannot be served until some point in the future.

Let:

I_t = Information used to make a decision at time t.

We can choose what action is used to make a decision. For example, we may choose to consider all aircraft and demands that are known now, and are actionable within a *planning horizon* T^{ph}. In this case, our information set would be written:

$$I_t = \left(R^A_{tt'}, R^D_{tt'}\right)_{t'=t}^{T^{ph}}$$

This means we are considering all aircraft and demands that are known now and actionable within the planning horizon.

3.3 Decisions with forecasts

The next step is to combine not only what we know, but a forecast of what we do not know. Let

$F^D_{tt'}$ = Forecast of demands to be served at time t' based on what we know at time t.

$F^A_{tt'}$ = Forecast of aircraft that are expected to be available at time t' based on what we know at time t.

We assume that these are “point forecasts”. For example, we might forecast that three aircraft will become available. The information set for making decisions now looks like

$$I_t = \left(R^A_{tt'}, R^D_{tt'}, F^A_{tt'}, F^D_{tt'}\right)_{t'=t}^{T^{ph}}$$

3.4 Decisions with value functions

We now introduce the idea of acting on an aircraft because we believe the action will add value (increase the benefit from revenue perspective), although we do not know exactly what it will do. This behaviour is achieved by estimating the *value* of an aircraft in the future (See [29][30] for the approximation of the value function).

This strategy is based on solving the problem as a dynamic program. Stated in general terms, if R_t is our (resource) state of the system, and if $V_t(R_t)$ is the value of being in state R_t, then we can use Bellman’s optimality equation to write:

$$V_t(R_t) = \min_{x_t \in X_t(R_t)} C(R_t, x_t) + E\left\{V_{t+1}(R_{t+1}(R_t, x_t)) \mid R_t\right\}$$

This is the classical statement of Bellman's optimality equation. Developed in the 1950's, it was quickly found to be a powerful theoretical device, but computationally impossible for most practical problems. The problem is the "curse of dimensionality" which means that as the dimensionality of the state variable R_t increases, the state space increases exponentially.

Computational results are typically limited to problems with fewer than a dozen dimensions. For airlift problems, the dimensionality can easily be in the thousands (or even millions). We have overcome this problem.

The equation above is formulated using what is known as a *pre-decision state variable*. This is the state of the system after new information becomes known, but before we make a decision. Our first step (which we already took in Chapter 2) was to formulate the state variable using the *post-decision state variable*, which is the state immediately after we make a decision. Formulated this way, the optimality equations look like

$$V_{t-1}(R_{t-1}) = E\left\{\min_{x_t \in X_t(R_t)} C(R_t, x_t) + V_t(R_t(R_{t-1}, x_t)) \mid R_{t-1}\right\}$$

We then drop the expectation and solve the problem for a sample realization of the information that arrives during time interval *t* (before we make a decision)

$$V_{t-1}(R_{t-1}, W_t(\omega)) = \min_{x_t \in X_t(R_t)} C(R_{t-1}, W_t(\omega), x_t) \quad V_t(R_t(R_{t-1}, W_t(\omega), x_t))$$

Finally, we have to replace the value function with a suitably chosen functional approximation:

$$\tilde{V}_{t-1}^n(R_{t-1}^n, W_t(\omega^n)) = \min_{x_t \in X_t(R_t)} C(R_{t-1}^n, W_t(\omega^n), x_t) \quad \bar{V}_t^{n-1}(R_t(R_{t-1}^n, W_t(\omega^n), x_t))$$

We have represented the equation assuming that we are at iteration *n*. Given an approximation $\bar{V}^{n-1}(R_t)$ at time *t* computed in the previous iteration, we can make a decision (call it x_t^n), which then determines the next state we will visit:

$$R_t^n = R^M(R_{t-1}^n, W_t(\omega^n), x_t^n)$$

The idea is that we simulate forward in time, visiting only the state that the approximation tells us to visit based on the decisions we make. The algorithm is sketched below.

Table 1: The approximate dynamic programming algorithm

Step 0: Initialize $\hat{V}_t^0(S_t)$, and pick an initial state S_0.

Set the iteration counter $n = 0$.

Set the maximum number of iterations N

Step 1: Set $R_0^n = R_0$, and select $\omega^n \in \Omega$

Step 2: (Forward pass)

Do for t = 1, 2,....,T-1:

Step 2.1 Let $\omega_t^n = W_t(\omega^n)$.

Step 2.2: Solve:

$$\tilde{V}_{t-1}^n(R_{t-1}^n, W_t(\omega^n)) = \min_{x_t \in X_t(R_t)} C(R_{t-1}^n, W_t(\omega^n), x_t) \quad \bar{V}_t^{n-1}(R_t(R_{t-1}^n, W_t(\omega^n), x_t))$$ Let

x_t^n be the optimal solution.

Step 2.3: Update the state of our system using:

$$R_t^n = R^M(R_{t-1}^n, \omega_t^n, x_t^n)$$

Step 3: If $n < N$, set $n \leftarrow n+1$ and return to **Step 1.**

It is this logic that makes the system truly an "optimizing simulator." We literally simulate the system forward in time (as we would with any simulation), but we do it iteratively, updating the value function at each iteration based on the information we have obtained. If we design our value function approximation properly, and if we are careful about how we update the value function (these steps are critical), we obtain optimizing behaviour in the sense that our solution will tend to get better over the iterations.

3.5 Decisions with patterns

A cost-model offers us the ability to handle problems with thousands of dimensions. However, the limitation is that we depend on the accuracy of the cost function to achieve desirable behaviours. If we do not like how the model is behaving, we have to change the costs and hope that this achieves the desired effect. Interestingly, simple rule-based systems are relatively easy to control (but they struggle with the challenge of working with high dimensional problems).

Our solution is to combine these two technologies. We are going to use the observation that when knowledgeable users object to the behaviour of the model, the objections can typically be represented as low-dimensional patterns. For example, "C-141's should be used to serve demands going to England" or "loaded C-17s should not be sent to airbases in Egypt." These are rules (when in this state, take this action) but they are expressed at a fairly aggregate level. They are also not expressed as hard constraints. Often, users like to encourage or discourage specific behaviours. As a result, these rules are not specific enough to replace a cost model, but they can be used to help guide a model.

We have found a way to merge the strengths of both techniques. We start with an engineering cost function of the sort expected by an optimization model. This would normally be written:

$$X_t^\pi(I_t) = \arg\min_{x \in \mathcal{X}} c_t x_t$$

Assume that an element of the decision vector x_t is given by x_{tad} which is the number of resources with attribute a that we act on using a decision of type d at time *t*. In real problems, the attribute vector *a* can be relatively complex (think of all the characteristics of an aircraft with a pilot moving a requirement to a destination). A *pattern* is the likelihood of a particular decision being used on a particular resource (note that we do not even consider the relationship between a decision and the state of the entire system; we restrict our patterns to relationships between decisions and the attributes of a *single* resource). However, it is very common for a pattern to be expressed at some level of aggregation of the attribute of the resource being acted on, as well as the decision itself. For example

1. Pattern: Aircraft CAN0123, a C-17 at airbase T2034 should move to airbase T2001 to pick up demand DEM051204 to move to airbase EGAV in England.
    a. Attribute: aircraft CAN0123 (which specifies all the characteristics of the aircraft).
    b. Decision: pick up DEM051204 to move to airbase EGAV in England.
2. Pattern: Demands out of T2001 should be served by CC130's's if they are going to England.
    a. Attribute: CC130s
    b. Decision: Moving orders out of T2001 going to England.

Let $\hat{a}$ be an attribute at some level of aggregation (CC130s in T2034, CC130s) and let $\hat{d}$ be a decision at some level of aggregation (moving a demand to EGAV, moving a demand to England). Let also x_{ad} be an aggregated value calculated using the vector x_{tad}. In the simplest case $x_{ad} = \sum_t x_{tad}$.

$$\rho^e_{\hat{a},\hat{d}} = \text{The fraction of time that a resource with attribute } \hat{a} \text{ should be acted on by a decision } \hat{d}.$$

The patterns $\rho^e_{\hat{a},\hat{d}}$ (target) are specified exogenously, either from historical activities or (more typically for a military setting) by a knowledgeable expert. We compare these to the patterns (at the same level of aggregation) that we derive from our model, which we compute using

$$\rho_{\hat{a}\hat{d}}(x) = \frac{x_{\hat{a}\hat{d}}}{\sum_d x_{\hat{a}d}}$$

Where $\rho_{\hat{a}\hat{d}}(x)$ is the frequency of using the decision $\hat{d}$ across the simulation horizon and $x_{\hat{a}\hat{d}}$ represent the number of decision with particular attributes (aggregation of values).

We incorporate patterns to our objective function by adding a penalty term for deviations from the pattern.

$$X_t^\pi(I_t) = \arg\min_{x \in \mathbf{X}} c_t x_t + \theta \sum_{\hat{a},\hat{d}} \left(\rho_{\hat{a}\hat{d}}(x) - \rho^e_{\hat{a}\hat{d}} \right)^2$$

These patterns can be quite simple. Since we use a cost function, it is not necessary that they fully specify the behaviour of the system. As a result, we can use patterns (defined on aggregated attribute vectors and decisions) that only have a few columns. We have found that in practice (this work has been used at a major trucking company and at two railroads) the patterns can be fairly simple. Furthermore, they may either be estimated from historical activities (as we have done at the trucking company) or through manually specified files. There can also be different types of patterns. For example, one pattern may specify the types of aircraft that we want to use for certain types of demands (e.g. those that involve oversized freight or passengers), while another might specify the size of aircraft (which may be an aggregation of types) that we prefer to land at certain airbases.

3.6 Optimization versus the optimizing simulator

There are several dimensions on which we can compare a classical optimization model to an optimizing simulator. These include:

- Solution quality – How do they compare in terms of providing the "best" solution?
- Realism – How well do they capture the real system? Can the model be calibrated to match actual performance statistics?
- Sensitivity to key policy variables – For planning purposes, does the system respond to the types of policy variables which are under study?
- Ease of use – How much preparation work is needed to produce a useful result?
- Speed – How much time does it take for the system to run?

It is important to emphasize that an optimizing simulator uses optimization. The major difference is that when looking at problems over time, a deterministic optimization model requires that we assume that all information is known in advance, whereas the optimizing simulator explicitly models the flow of information. Even more than this, the central feature of the optimizing simulator is that it models *the organization and flow of information and decisions.*

We begin our discussion by focusing on solution quality. Approximate dynamic programming has been tested on a range of deterministic and stochastic multicommodity flow problems. These are simpler than the problems faced in the military airlift arena, but we can solve them optimally (although not necessarily obtaining integer solutions) using a commercial solver. For the deterministic problems, we were able to find optimal solutions if we ignored the integrality requirement (the approximate dynamic programming solutions are always integer). The objective function produced by the approximate dynamic programming algorithm expressed in terms (as a percent) of the optimal deterministic solution (ignoring integrality) is shown in Figure 6. Most of the solutions are within one or two percent of the optimal. What is not known is the gap associated to the integrality constraint relaxation.

Experiments on datasets with stochastic demands were then conducted. Here, we compare with a rolling horizon procedure based upon future event point forecasts. All the results are reported as a percentage of the optimal solution after all demands are known (this is referred to as the posterior bound). The approximate dynamic programming procedure outperforms the rolling horizon procedure, except for one case (See Figure 7).

A point by point comparison of classical optimization and the optimizing-simulator is given in the Table that follows.

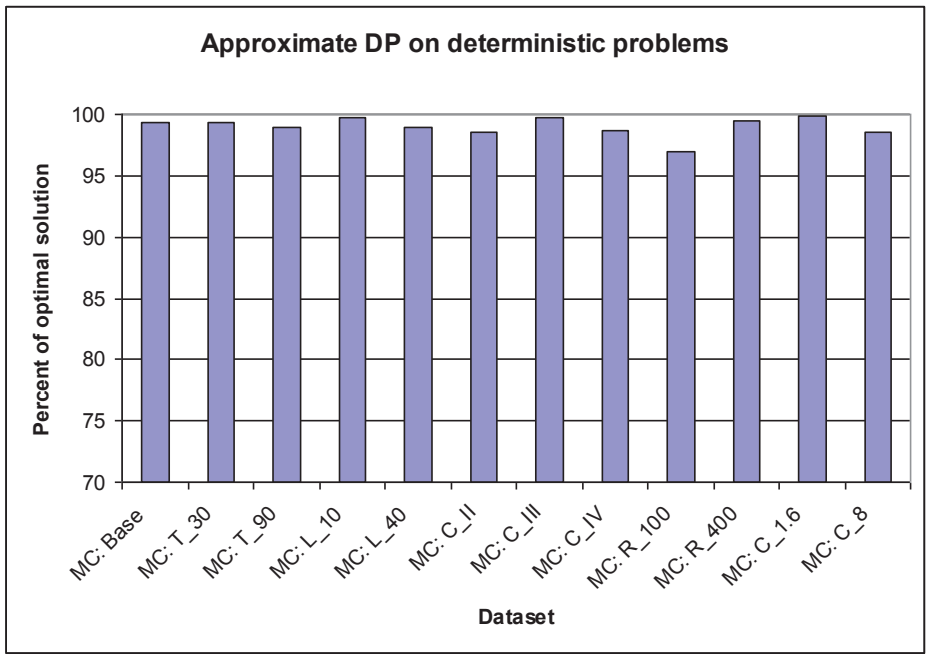

Figure 6: Total costs as a percent of the optimal for deterministic multicommodity flow problems

.

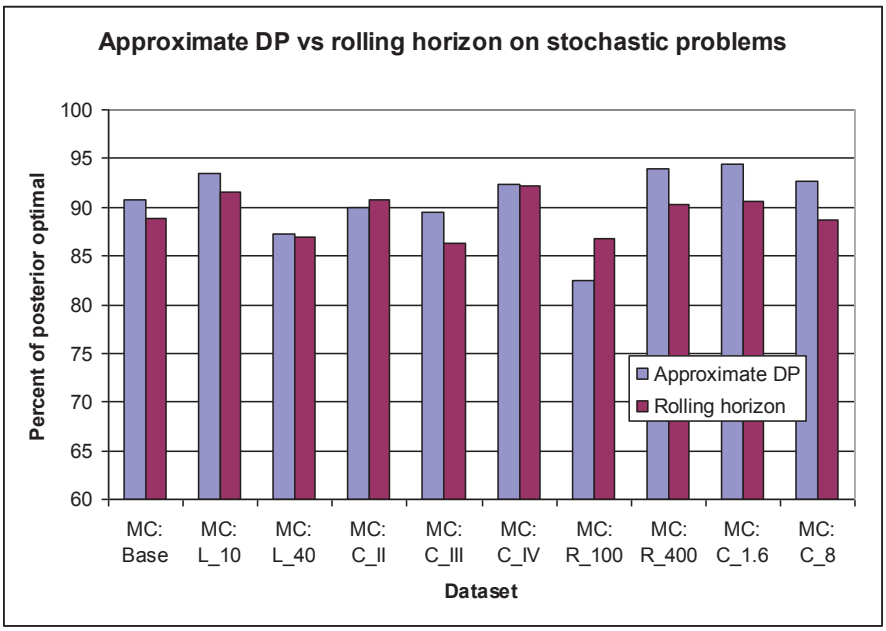

Figure 7: Total costs from approximate dynamic programming and rolling horizon procedures as a percent of the posterior optimal solution

Table 2: Comparison of classical optimization and the optimizing-simulator

Comparison of optimization versus the optimizing-simulator		
Issue	Optimization	Optimizing-simulator
Solution quality	For a well-defined objective function, if there is no uncertainty and we accept how the optimization model represents the problem, a pure optimization model is almost always going to provide a lower cost solution. Exceptions arise when the solution has been restricted to produce faster run times.	When using approximate dynamic programming, an optimizing simulator has been shown to produce solutions within one percent of the optimal for single and multicommodity flow problems. This degrades somewhat when there is a strong sequencing component to the problem.
Solution quality under uncertainty	Deterministic optimization requires using a point forecast of future events. These can be "resampled" as we move forward in time (a rolling horizon simulation). Solution quality is generally good but clearly suboptimal.	Approximate dynamic programming can produce provably optimal results in special cases, and very near optimal solutions in others. Optimal solutions do not exist for more realistic problems under uncertainty, but ADP consistently outperforms deterministic optimization using a point forecast.

Realism: representing resources:	Column generation models can represent resources using multiattribute vectors which provides for a high level of realism, but there is a limit to this (it is not the case for all models or problems). To produce rapid solutions, multicommodity flow codes almost always have to dramatically simplify the representation of resources.	DRTP model allows resources to be represented at an arbitrarily high level of detail. Optimizing simulator can represent resources in the future at a lower level of detail, something that optimization models cannot do. This allows us to use more detail to represent a resource.
Realism: modeling uncertainty	Deterministic optimization models cannot model the flow of new information that arrives during the planning. They can be solved on a rolling basis – this is identical to an optimizing simulator using a forecast.	Optimizing-simulator can easily handle a broad range of dynamic information processes. With approximate dynamic programming, solution quality matches or outperforms solutions produced by provably optimal stochastic programming
Model calibration	Deterministic optimization models must be calibrated by manipulating costs and constraints.	Optimizing simulator can also be calibrated using costs and constraints, but can also be calibrated using low-dimensional patterms.
Sensitivity to policy parameters	Optimization models have the feature that it tends to respond very smoothly to changes in inputs (more aircraft, more demands, and changes in capacity). But it can only capture changes to features that it is able to model. These are restricted to the technology used by optimization to represent the problem.	An optimizing simulator tends to bounce from one iteration to the next, much as any simulation model would. Small changes in inputs may have to be analyzed statistically. The exception is that it obtains gradient information just like the dual variables we can obtain from an optimization model. The analysis group at the airlift mobility command prefers simulation over optimization because it can model almost anything.

Ease of use	If an analysis problem can be captured by an optimization model, using optimization can be quite easy because the algorithms are relatively mature and stable. Calibration can be difficult, but the inherent intelligence of optimization tends to produce realistic solutions with relatively little tuning.	Cost-based optimization models usually produce good solutions very quickly (true of both technologies). Optimizing simulator offers greater flexibility for handling new issues, but this generally requires custom programming. Optimizing simulator offers the use of patterns to improve solution quality without changing code, but the type of pattern has to be coded into the model. The inherent noise in an optimization simulator can require the use of statistical techniques for analysis (as with any simulation package).
Speed	With sufficient engineering, optimization models can be reasonably fast, but they struggle with long planning horizons as well as flexibility in when a demand is served. Adding dimensions such as pilots can greatly complicate an optimization model, but ultimately the speed depends largely on the level of engineering put into the project.	An optimizing simulator can be run from a cold or warm start. From a cold start, it can require as much or more time than an optimization model for initial training of the value functions. Once these are trained, many analyses can be done with a single pass, which can be much faster than an optimization model.

4 Measuring the value of increasing information

A series of experiments were run using an unclassified dataset from the US Airlift Mobility Command. These experiments were designed to test the value of increasing the amount of information available to make a decision. The first set of experiments compared rule-based logic (emulating the logic currently used in the AMOS simulator currently used by the analysis group at Scott Air Force Base in US) to various forms of cost-based logic, but excluding the logic that included patterns. The following five classes of information sets (and decision functions) were used:

1. Rule-based logic – The system would take the first demand that had to be filled, and try to match it with the first aircraft available to move the demand. If the assignment was feasible, the assignment would be made.

2. Cost-based logic, single demand – We would take the first demand that needed to be served, develop a list of possible aircraft sorted on the basis of cost, and choose the one with the least cost.

3. Cost-based logic, multiple aircraft and demands, but restricting to those which were both known now and actionable now. A linear programming package was used to solve the assignment problem, but ignoring the possibility of assigning an aircraft now to a demand that is known but cannot be served until the future.

4. Cost-based logic, with aircraft and demands that are actionable within a planning horizon. An aircraft that is actionable in the future might be assigned to a demand that is actionable in the future, as long as these occurred within a specified horizon.

5. Decisions using value functions. Here, we use value function approximations to estimate the value of an aircraft in the future.

The effect of simulation can be measured in three different ways: the costs generated, throughput (how fast the freight moves), and how well the system matches desired patterns. These are summarized below.

4.1 The effect of information on costs

The different policies (with different amounts of information) were first compared on the basis of the total cost function. This included transportation costs, penalties for late deliveries, and repair costs. We modeled the possibility that certain aircraft might have higher repair costs at certain airbases. But these costs are only incurred if the aircraft actually fails at the airbase.

The results of this study are shown in Figure 8, demonstrating that increasing the information can reduce the costs of the policy. Note that the repair cost component virtually disappears when we use value functions that capture the cost of putting certain types of aircraft into certain types of airbases.

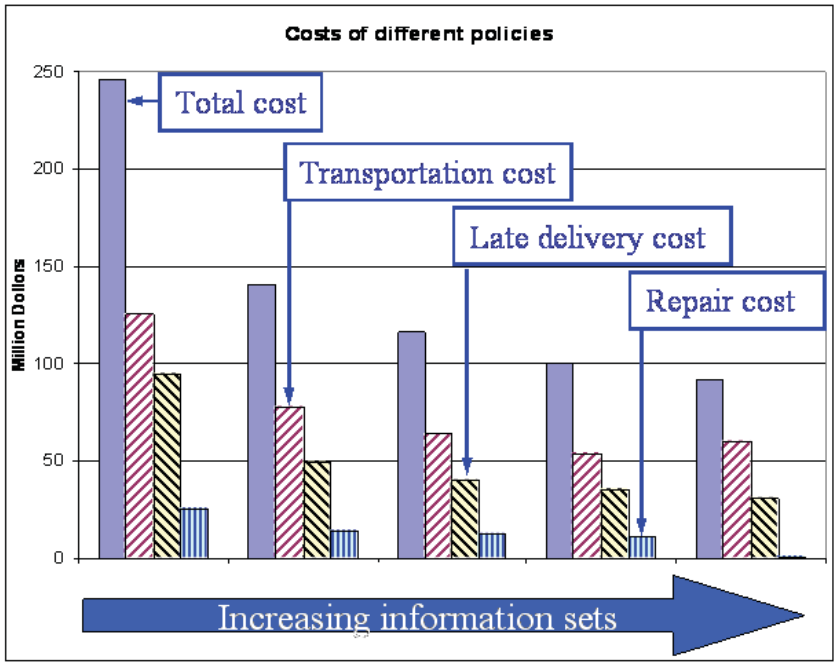

Figure 8: Costs produced by simulations for each of the five information sets and decision rules

4.2 The effect of information on throughput

We next compared the results based on total throughput. This is the measure that is of greater interest to the military. Figure 9 shows the total pounds delivered as a function of time for each of the decision rules. The left-most curve is the cumulative pounds as they enter the system. This would be the throughput curve if every pound were immediately delivered (infinite capacity aircraft flying at infinite speed). The area between this curve and an actual throughput curve is a measure of the delay incurred as a result of system capacity. While there is no guarantee that the throughput would get better, we did find, as shown in Figure 10, that the throughput did indeed get faster as we increased the information content of a decision.

It is important to emphasize that we are not seeking optimal solutions. Rather, we are simulating decision functions with increasing degrees of intelligence. Not surprisingly, there are decreasing returns as we add intelligence, and it is entirely possible that we may decide that some policies are unrealistically good.

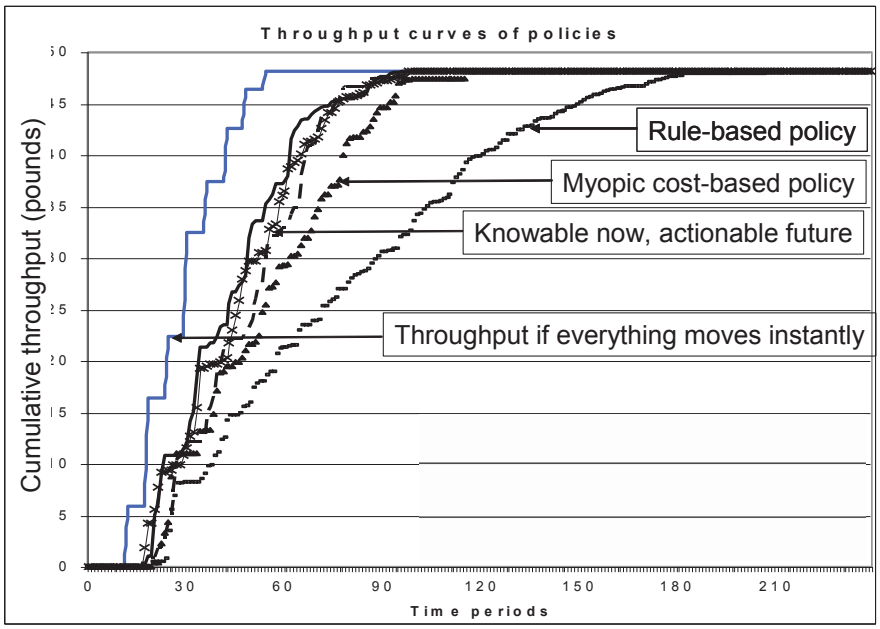

Figure 9: Cumulative throughput produced by each decision function and information set

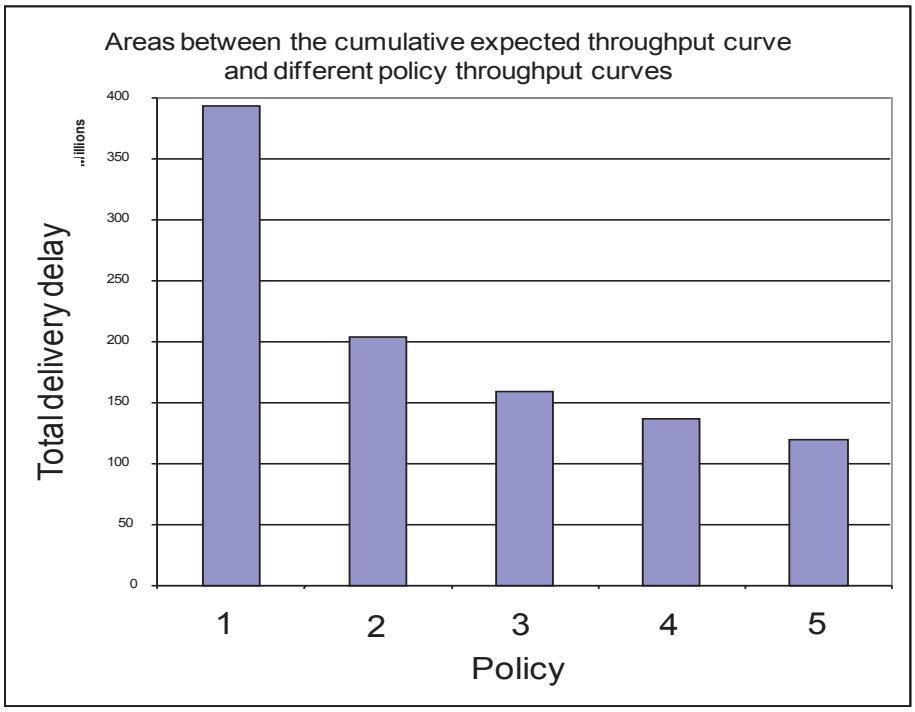

Figure 10: Total throughput curves for each of the information sets and decision functions

4.3 Matching patterns

It is often the case that someone knowledgeable in operations will say that a model should do more or less of an activity to be more realistic. We tested the ability of the model to match a pattern which specified the percent of flow through a particular airbase which should be C-141's. If the pattern is ignored ($\theta = 0$), then 18 percent of the flow through the terminal consists of C-141's. Now assume that we wish the percentage to be some fraction ρ, which we may vary between 0 and 1. We ran the model with increasing values of θ. The results are shown in Figure 11. With $\theta = 0$, the fraction is 18 percent for all values of ρ. As θ is increased, the model more closely matches ρ.

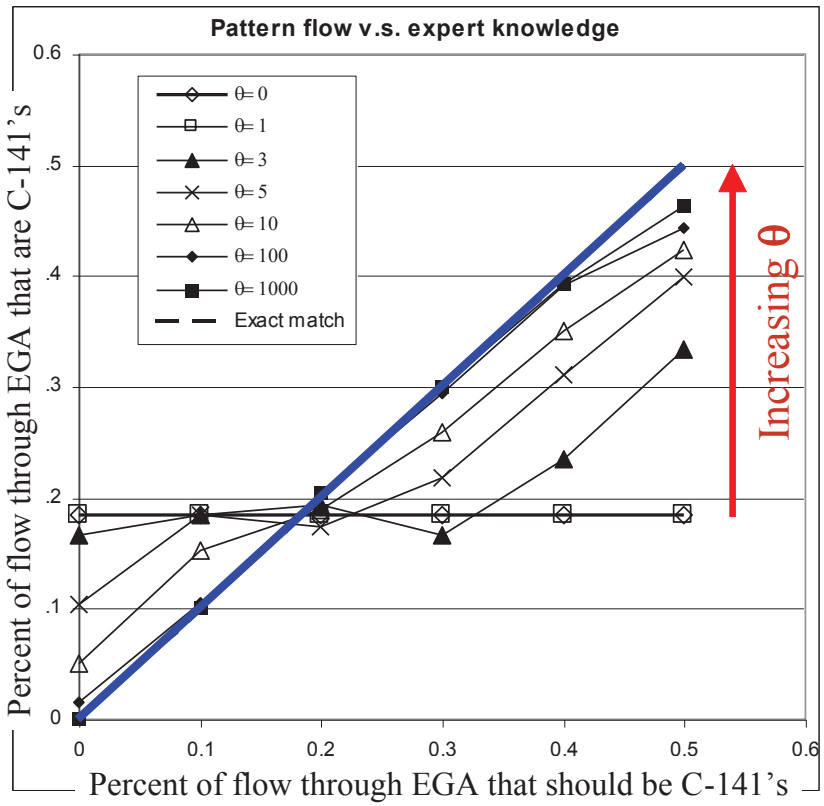

Figure 11: Percent of flow through airbase EGA that was C-141's versus the desired percentage for different values of theta

5 A model of the CF airlift problem

In this chapter, we provide a more complete model of exactly how we modeled the CF airlift problem. Our model is restricted by the available data, and throughout the presentation we indicate how the model could be generalized using the DRTP modeling framework and the optimizing-simulator algorithmic framework.

5.1 Resources

For this initial demonstration project, we modeled only two resource layers:

A: Aircraft

D: Demands, representing loads of freight to be moved.

Aircraft were modeled using:

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} \text{Aircraft type} \\ \text{Location} \\ \text{Time of availability} \\ \text{Loaded/empty} \\ \text{Equipment status} \end{pmatrix}$$

We included aircraft type (a_1) although in the dataset, all the aircraft were identical. The "location" field is not necessarily the actual physical location of the aircraft (which we can easily include). Here, it means the next decision point, where "time of availability" is when we are expected to be at this decision point. An aircraft could be at field AAA at time t = 100 (minutes), but if the decision is to send it to BBB, with an expected arrival time of t = 192, then we would show the location as BBB, with time of availability of 192. The equipment status field captures whether the aircraft needs repair or not.

In addition to aircraft type, it would be normal to also include an attribute for configuration. The deterministic optimization did not seem to allow for enroute changes in configuration. We would have no difficulty adding this as a feature.

The time of availability is what we refer to as the *actionable time*. A plane that is enroute to a destination will generally not be actionable until it arrives. However, we can model enroute changes in plans by letting every time period be an actionable time. In this case, "location" is not the ultimate destination (this would have to be captured with a new attribute) but instead is the expected location at the time when the next decision can be made. We could also add attributes such as maintenance status, with random changes in this status at the end of each flight.

To streamline the notation, we represent demands using the attribute vector b:

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \\ b_{10} \\ b_{11} \\ b_{12} \end{pmatrix} = \begin{pmatrix} \text{Origin} \\ \text{Destination} \\ \text{Required a/c type} \\ \text{Brief time} \\ \text{Load time} \\ \text{Unload time} \\ \text{Debrief time} \\ \text{Depature time window} \\ \text{Arrival time window} \\ \text{Priority} \\ \text{Knowable time} \\ \text{Type} \end{pmatrix}$$

The knowable time (b_{11}) represents the time at which the demand becomes known. Strategic demands are all known in advance and generic demands become known during the simulation. Since data was not available for this element, we ran simulations with different assumptions of when generic demands became known. In a full-scale system, the type of demand might include whether it is a strategic (intertheatre), tactical (intra-theatre), refueling and search-and-rescue.

We note that the concept of departure and arrival time windows is a data representation that is very common in deterministic optimization models. In a stochastic dynamic world, it is more natural to have the beginning of a departure time window (the earliest time the cargo will be available to be moved). Different forms of uncertainty make the end of a time window soft. For example, we may assign an aircraft to move a cargo with the full expectation that it can arrive and depart within the window, but an equipment failure may violate this. As a result, it is more common to think of the beginning of the window as a hard constraint (the demand physically is not ready to be moved), whereas there is a penalty for moving it after the end (but we allow for weather delays and equipment failures). Similarly, arrival time windows tend to be goals rather than constraints. Normally, once an aircraft takes off, it moves as quickly as possible to the destination. Again, the beginning of the arrival time window may be a hard constraint (the destination does not have the physical resources to handle the aircraft before this time), but the end of the time window is a goal rather than a constraint. Time windows serve as useful pruning rules for deterministic models (in fact, column generation methods become quickly impractical in the presence of wide time windows). For consistency, we treated the time windows as hard constraints. If we arrive before the beginning of a window, we wait. If we cannot arrive before the end, we assume the demand cannot be served.

5.2 Decisions

We assumed that aircraft were an *actionable resource* (decisions could act on aircraft to change its attributes), whereas demands were *passive* (an aircraft can pick up a demand, but we cannot act on a demand by itself other than to “do nothing”).

If we wish to allow the leasing of aircraft from outside the fleet for specific moves, then we could move a load without using one of the aircraft in our fleet.

We model decisions using:

D^L = Set of decisions to move an aircraft to a specific location. The set D^L is the set of locations we can move to.

D^D = Set of decisions to ``serve a demand" of a particular type. The set D^D is the set of all demand types.

d^ϕ = "Do nothing" decision.

$\mathrm{D} \equiv \mathrm{D}^L \cup \mathrm{D}^D \cup d^\phi$

The set D^L captures our ability to move to a location in anticipation of a demand that we do not yet know about. The set D^D may require moving to another location, but the decision is to cove a specific known demand. D^D could be the set of demands we know about at time t (in this case, we would write it D_t^D) but we have found it more convenient from a notational perspective to let D^D be demand types, where a type is characterized by a vector of attributes. For our discussion, it is possible to think of D^D as being a specific demand (the attribute vector is so specific that it is virtually impossible to have two separate demands with the exact same attributes).

We have modeled an element $d \in \mathrm{D}^D$ as being either a single demand, or a set of two or three demands which the dataset has already linked as potentially being served together. The same demand can be covered by more than one decision (we can decide to serve a demand by itself, or as part of one pair of demands or a separate group of three demands). We introduce a linking constraint to restrain the model so that it cannot cover the same demand twice.

Over the course of a simulation, a single aircraft can, of course, cover a number of demands in sequence. In our initial implementation, the *decisions* to cove these demands are made one at a time. That is, at a given time *t*, we will decide to assign an aircraft to at most one demand at a time (with the exception of demands that are already grouped into an external dataset). It is also possible to solve a more difficult problem where we plan the movement of an aircraft through a sequence of demands that are all known at time *t*. This is exactly what is done in a column generation (or multicommodity flow) model. Thus, a decision *d* can be a sequence of several demands (a "column"), where we would use a set partitioning formulation to determine which of the "columns" (*d)* we should choose. Such an approach would provide higher quality solutions for a deterministic dataset. But in practice, it does not always make sense to plan a schedule for several days in the future if new information (new demands, weather delays, equipment failures) are going to disrupt this schedule. In addition, it makes the algorithm considerably slower.

It would be quite easy to model other types of demands. For example:

D^M = Set of decisions to "modify" an aircraft

The set D^M could represent a decision to change the configuration, repair an aircraft or refuel an aircraft.

5.3 System dynamics

System dynamics describes how the state of our system evolves over time. We model physical events in continuous time (for example, departures and arrivals can occur at any time), but we model decisions in discrete time (we used six hour time steps for our experiments here). For this reason, we have to model the evolution of what we know about the system in discrete time. The state of our system is given by

$$R_t = \left(R_t^A, R_t^D\right)$$

Where

$$R_t^A = \left(R_{ta}^A\right)_{a \in \mathrm{A}}$$

$$R_t^D = \left(R_{tb}^D\right)_{b \in \mathrm{B}}$$

In our software, we model an aircraft as an attribute vector *a* that changes over time. In our mathematical model, we prefer to use the notation R_t^A, which makes the description of the algorithm easier.

The attribute vector *a* of an aircraft changes because of *decisions* and *exogenous information.* We let $d \in \mathrm{D}_a^A$ represent a decision acting on a single aircraft (with attribute *a*), and we let x_t be the vector of decisions acting on all the aircraft represented by R_t^A. It is easiest to focus on the effect of a decision *d* acting on a single aircraft. Earlier, we modeled this effect using the *modify* function, which we wrote as follows:

$$M(a_t, d_t, W_{t+1}) \rightarrow (a_{t+1}, c, \tau)$$

Here, the attribute a_t takes the form of a *pre-decision attribute vector*, which is to say the attribute vector just before we make a decision. We may let a_t^x be the attribute vector immediately after we make a decision, but before any new information has arrived. Finally, a_{t+1} is the attribute vector after the information W_{t+1} (weather delays, equipment failures) is included. Below we illustrate an initial attribute vector, the attribute after a decision is made (with a new location and estimated time of arrival, which is τ added to the current actionable time of the aircraft) and finally the attribute after the plane arrives (which was delayed enroute, which includes updates to the original actionable time).

$$a = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} \text{CC130} \\ \text{AAA} \\ 2005:07:20:09:30 \\ \text{Empty} \end{pmatrix} \quad \begin{pmatrix} \text{CC130} \\ \text{BBB} \\ 2005:07:20:11:47 \\ \text{Loaded} \end{pmatrix} \quad \begin{pmatrix} \text{CC130} \\ \text{BBB} \\ 2005:07:20:12:22 \\ \text{Empty} \end{pmatrix}$$

Initial attribute t=2005:07:20:06:00 | Attribute after decision t=2005:07:20:06:00 | After new information t=2005:07:20:12:00

A number of calculations take place in the modify function. For example, consider the decision to assign an aircraft in one location to a demand in another. To determine feasibility, the modify function would have to perform the series of calculations indicated in Figure 12. Before loading, the aircraft (and crew) have to pass through a brief time followed by the aircraft loading time. This may take place before the start of the departure time window, so that the actual departure time occurs within the time window.

The movement from one airbase to another may have to progress through one or more intermediate airbases for refuelling. We assumed an aircraft could be refuelled at any of the locations we were given, and found the shortest path through these points. In practice, it is more likely that there are only a small handful of eligible refuelling stations. The modify function ensures that it is possible to arrive within the arrival time window (if we arrive early, the aircraft will simply wait).

The modify function can handle a vast range of operational issues, at the level of one aircraft at a time. The concept is identical to what is happening within a column generation routine, with the exception that we do not plan the entire path of the aircraft over the entire simulation (this is the logic used in column generation). It is exactly this property that makes our methods so much faster than a column generation method.

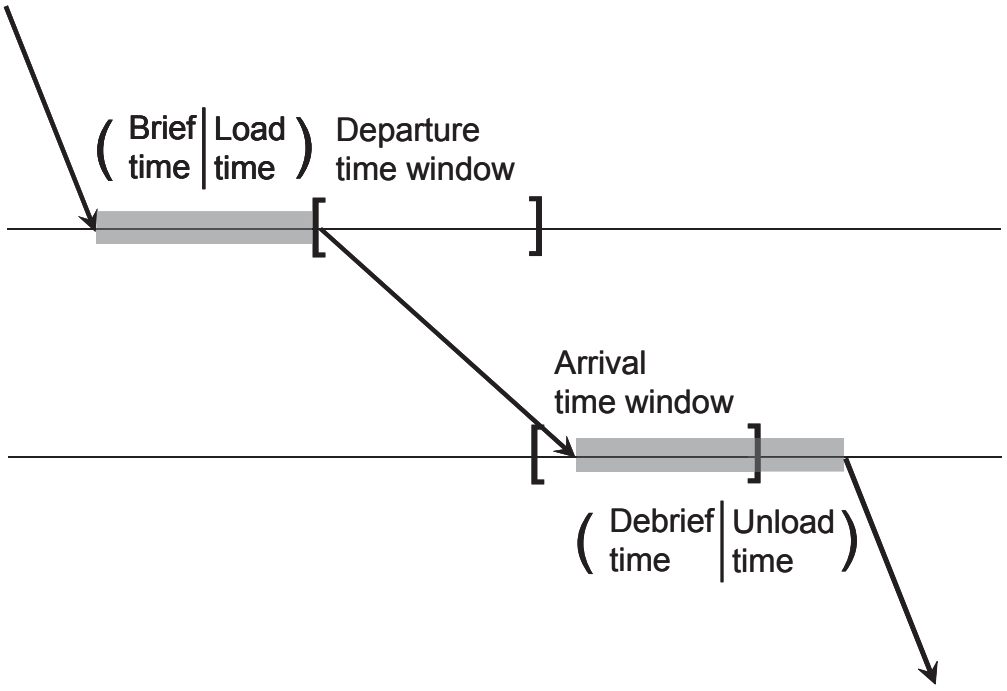

Figure 12: Relationship of brief and load times to departure and arrival time windows

The most important source of exogenous information is new customer requests. We model these using:

$\hat{R}^D_{ta}$ =Number of new demands that are first learned at time t with attribute a.

Within the attribute vector a is the departure time window that determines when the order can first be moved. The evolution of our resource vector is given by

$$R^A_{t+1,a'} = \left(\sum_{a \in A} \sum_{d \in D} \delta_{a'}(a_t, d_t) x_{tad} \right) + \hat{R}^A_{t+1,a'}$$

The first term is the number of aircraft with attribute *a'* immediately after we have made our decisions, while the second term captures the effect of new information. Here:

$$\delta_{a'}(a_t, d_t) = \begin{cases} 1 & \text{If decision } d_t \text{ acting on } a_t \text{ produces a resource with attribute } a' \\ 0 & \text{Otherwise} \end{cases}$$

is modeled deterministically. That is, *a'* is the attribute vector before any new information arrives.

5.3 Costs and rewards

For this initial demonstration, we used a fairly simple cost structure:

"Hard" costs:

Transportation cost: $50 per hour of travel time.

Rewards for moving a demand: This was set to $2000 plus a coefficient times the priority (priority = 1 is the lowest priority, 8 is the highest). We used a coefficient of $10 (this is primarily a tie-breaking bonus).

"Soft" costs

At time t, it is more important to serve demands with shorter departure time windows. Let t be the current time, and t^{end} be the end of a time window. We subtract $\$20\,(t - t^{end})$ from the value of a demand, making demands with a lot of remaining time less valuable.

If two demands are served as part of one movement, this is preferred over serving them individually. We add a $30 bonus for serving requests together.

We do not allow late pickups or deliveries, but it would be very natural to allow these with a penalty.

5.4 The decision function

At any one point in time, we can assign aircraft to demands, or as movements to physical locations (see Figure 12). A movement to a physical location has the effect of creating an aircraft with a vector of attributes in the future. As we pointed out earlier, we can also consider decisions that modify the attributes without moving the plane to another location (e.g. reconfiguring the aircraft).

From each aircraft node (these nodes may have zero, one or more than one aircraft) we either generate an assignment arc (to a demand) or an arc to a node from which we have a series of parallel arcs which model a piecewise linear value function. These parallel arcs capture the marginal value of an additional aircraft with a particular set of attributes. We note that when we model the value of a particular type of aircraft (for use in the value functions) we typically do so

at a more aggregate level (that is, we use a simpler vector of attributes). For these experiments, the attributes were already quite simple, so additional aggregation was not necessary.
The problem depicted in Figure 13 is a linear program (actually, an integer program).

Mathematically, the problem is given by

$$x_t^n = \max_{x_t \in X_t(S_t} \sum_{a \in \mathrm{A}} \sum_{d \in \mathrm{D}_a} c_{tad} x_{tad} \quad \sum_{a' \in \mathrm{A}} \bar{V}_{ta'}^{n-1}\left(R_{ta'}^x(x_t)\right) \tag{1}$$

Subject to

$$\sum_{d \in \mathrm{D}_a} x_{tad} = R_{ta}^n \tag{2}$$

$$\sum_{a \in \mathrm{A}} x_{tad} \le R_{tb_d}^D \qquad d \in \mathrm{D}^D \tag{3}$$

$$\sum_{a \in \mathrm{A}} \sum_{d \in \mathrm{D}_a} \delta_{a'}(t,a,d) x_{tad} - R_{ta'}^x = 0 \tag{4}$$

$$x_{tad} \ge 0 \tag{5}$$

$\bar{V}_{ta}^{n-1}\left(R_{ta'}^x(x_t)\right)$ is a piecewise linear function of the total flow into attribute node *a'*. Equation (2) is our flow conservation constraint. Equation (3) applies only to decisions $d \in \mathrm{D}^D$ that represent serving demands, where for each $d \in \mathrm{D}^D$, there is a corresponding demand type $b_d \in \mathrm{B}$ (where B is the set of all possible demand types). The constraint tells the model that we can only cover an order once. The entire problem is being solved for a particular sample realization ω^n (representing the n^{th} sample realization of all random variables).

Solving it returns a vector x_t. If $x_{tad} = 1$, then we are acting on an aircraft with attribute *a* with decision *d*. Often, these linear programs are networks, which means a linear programming solver naturally returns an integer solution. When there are different aircraft types, or if there are airbase capacity constraints, then resulting linear program is not a network, but has a near-network structure so that the integer program is quite easy to solve.

It is important to understand how constraints are handled in this setting. When we solve a decision problem, the only constraints are flow conservation, capacity constraints (two aircraft cannot pick up the same cargo; a tanker cannot perform mid-air refuelling for two aircraft at the same time; an airbase may have a limit on the number of aircraft that can be on the ground at the same time), and nonnegativity. It is common for optimization models to handle other issues using the language of "constraints" (time windows; travel times). We handle these issues in the modify function. We do not think of the modify function as a "constraint." In the simulation literature, this is often called a transition function. We use "constraints" to ensure valid decisions at a point in time, and we use a transition function (such as the modify function) to handle the dynamics of the system over time.

Once we have an optimization decision vector x_t, we now have to update the state of the system. If $x_{tad} = 1$, then this means we are acting on an aircraft with attribute a_t with decision type *d* to

produce an aircraft with attribute a_t' (this is the post-decision attribute vector). We can then simulate additional information (weather delays, equipment failures) to produce the attribute vector a_{t+1}.

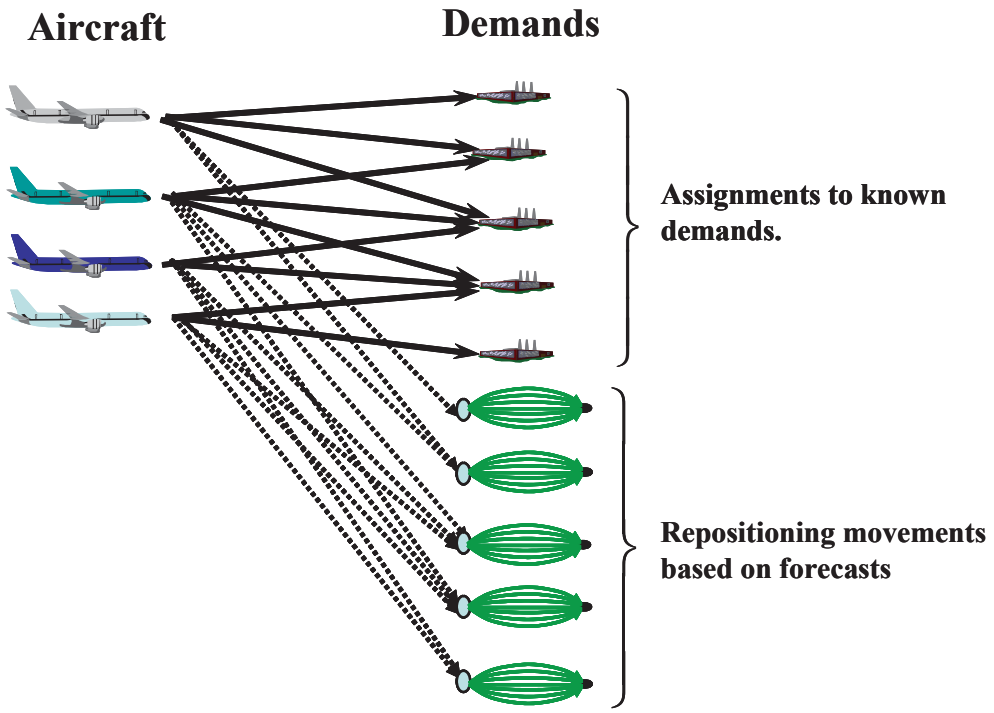

Figure 13: Decision problem, assigning aircraft to demands or to locations

It is important to emphasize that the discrete time steps *t, t+1, ...,* represent the times at which we make decisions, and not when physical events happen. We ran our simulations using time steps of 360 minutes. We might decide at t=360 to assign an aircraft which is available (actionable) at t=420. If the movement requires 200 minutes, then the decision at t=360 to assign the aircraft would produce a departure at 420 with a subsequent arrival to the destination at t=620. If we wish to introduce random travel times (due to weather delays, we might then randomly sample a weather delay of 32 minutes (for example), producing an arrival time of t=652.

5.5 Updating the value function

There are several strategies for updating the value function. Here we illustrate only the simplest strategy.

When we solve the linear program, in addition to obtaining the optimal decision vector x_t^n, and we also obtain dual variables $\hat{v}_{ta}^n$ for the resource conservation constraints (2). These dual variables are an approximation of the marginal impact of the resource vector R_{ta}^n. We illustrate the updating strategy when the value function is linear in the resource vector:

$$\bar{V}_t^{n-1}(R_t^x(x_t)) = \sum_{a \in \mathrm{A}} \bar{\mathbf{v}}_{ta}^{n-1} R_{ta}^x(x_t)$$

The marginal value of resources of type *a*, given by $\bar{v}_{ta}^{n-1}$, is updated using

$$\bar{v}_{ta}^{n} = (1 - \quad^{n})\bar{v}_{ta}^{n-1} + \alpha^n \hat{v}_{ta}^{n}$$

where α^n is a stepsize between 0 and 1 which has to have certain properties. For example, one formula is of the form

$$\alpha^n = \frac{\bar{\alpha}}{\bar{\alpha} + n - 1}$$

In our work, we will use a linear value function approximation when the resource attribute vector is moderately complex, since for these problems we typically have $^{x}_{ta}(x_t)$ (the number of aircraft with type *a*) on the order of 0, 1 and 2. For the experiments in this project, the attribute vector is very simple, and larger resource values are possible. For this reason, we used a piecewise linear value function approximation. There are a number of methods for updating this function. We do not provide the details of how we update the value function approximation. For more information on this, details are provided in the book [31].

When we are modeling simple assets, we will enumerate the attribute space, solving the optimization model (1)-(5) (and in particular, the flow conservation constraint) for all $a \in \mathrm{A}$, even when $R_{ta}^n = 0$. For more complex problems, we cannot enumerate the attribute space. When this happens, we solve the optimization problem for a subset of attributes that we denote (at iteration *n*), A^n which expands from one iteration to another. One problem we face is that we may wish to consider acting on an aircraft with attribute *a* with decision *d* producing an attribute *a'*, but we have not created a value function approximation $\bar{V}_{ta'}^{n-1}$. Techniques for solving this problem are described in [31].

6 Results of experiments with Canadian military air mobility datasets

A series of simulations were run using Canadian Air Mobility data. The geographical coverage is shown in Figure 14, with a closeup of eastern Canada in Figure 15. Noting that loaded movements are shown in blue while empty movements are in red, we see that almost every loaded move is preceded by an empty move, some of which require an aircraft to move over long distances.

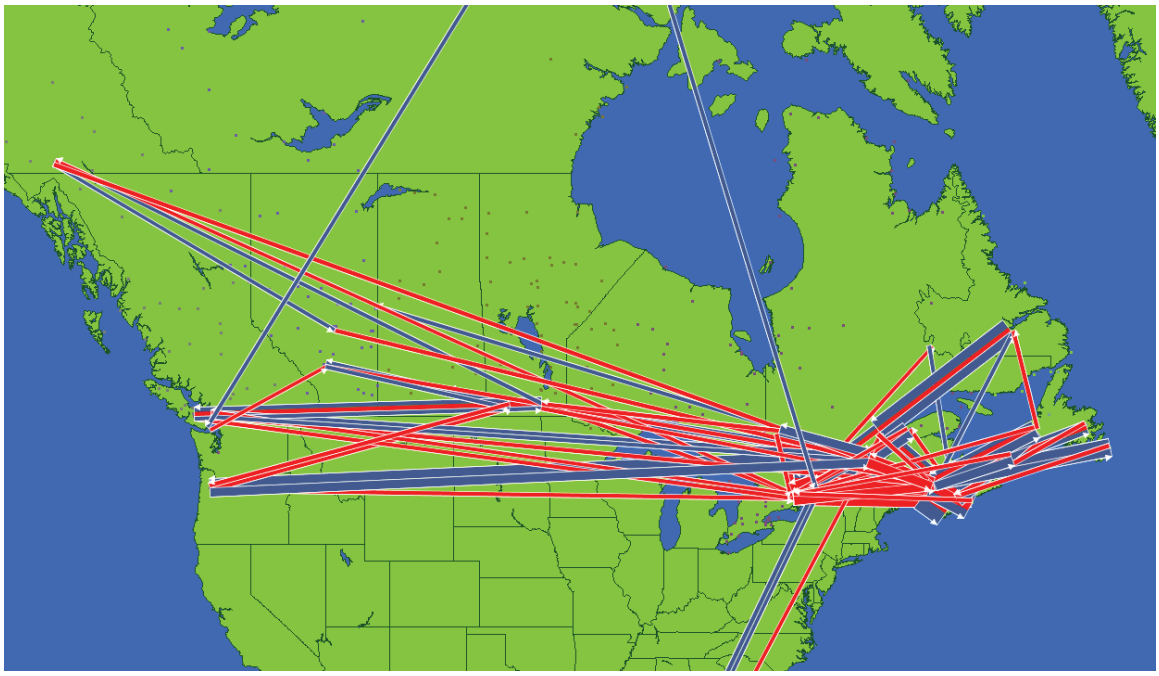

Figure 14: Airlift flows (blue = loaded, red = empty)

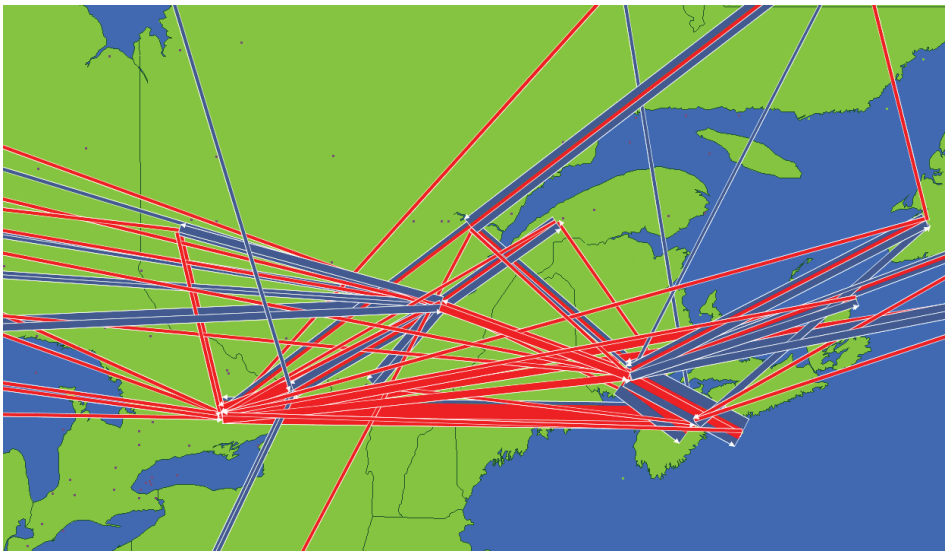

Figure 15: Flows over eastern Canada

6.1 PILOTVIEW diagnostic tools

The PILOTVIEW[1] diagnostic system allows us to see what the model is doing as well as to drill in and examine decisions at a high level of detail. The "PILOTTOUR" module allows us to see detailed aircraft and demands. We can see not only what assignments the model made, but also which were considered (but not chosen), and which were not considered at all (when no link is generated). Figure 16 shows a closeup of individual assignments of aircraft to demands at a point in time.

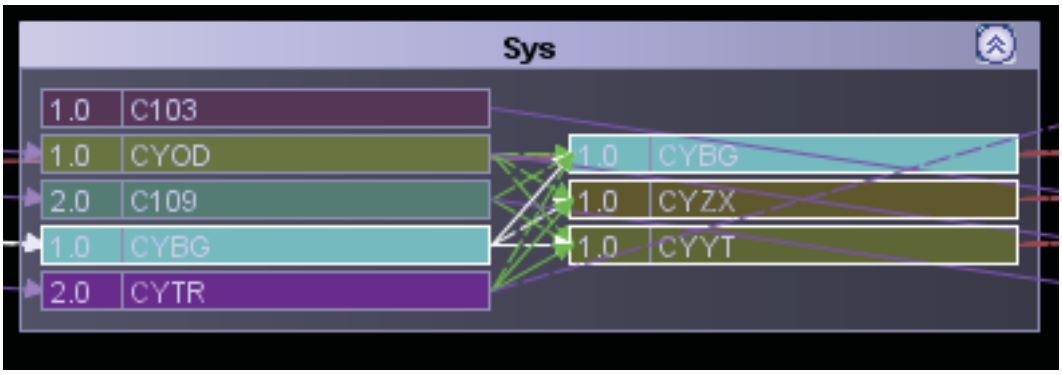

Figure 16: Detailed depiction of aircraft and demands, and all possible assignments

[1] PILOTVIEW is a diagnostic tool (developed at Princeton University) allowing people to understand what a model is doing, and why.

Figure 17 shows how PILOTTOUR allows us to follow the paths of vehicles over time. This capability allows us to track the path of an aircraft over the entire simulation.

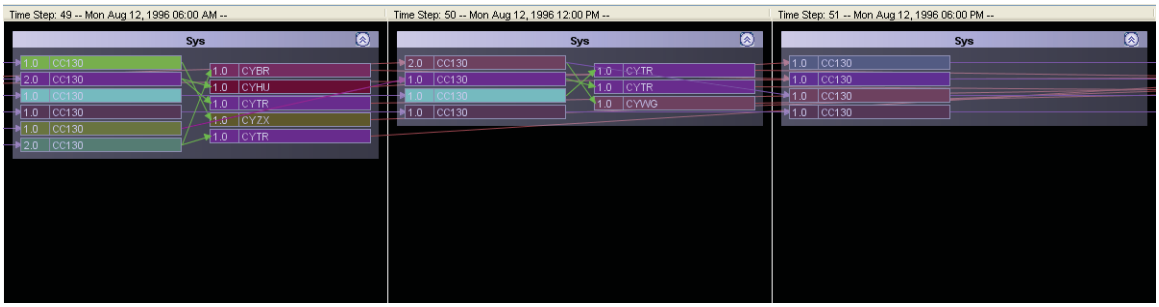

Figure 17: Following assignments over time

We also can click on an individual aircraft or demand and see detailed information about each asset, as shown in Figure 18. This drill-down capability allows us to see not only the attributes of the aircraft, but also other information that is derived within the optimizing-simulator (such as dual variables value functions). For the demands, we have access to information such as the type of aircraft (and configurations) that can cover a demand, the departure and arrival time windows, and the priority.

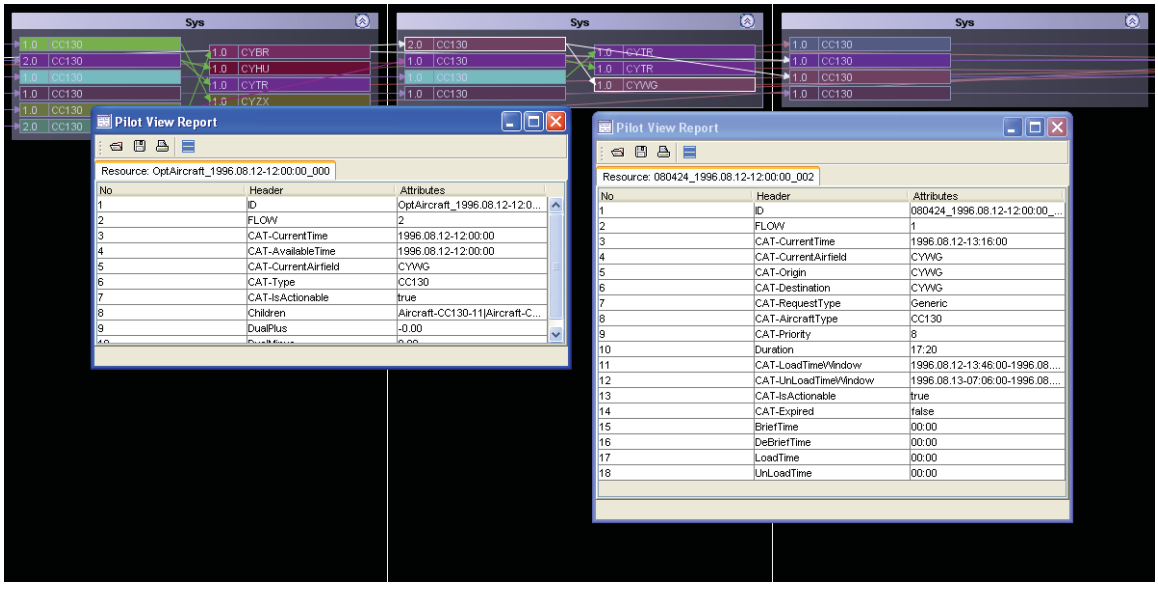

Figure 18: Drill down capabilities for aircraft and demands

In addition, we can click on a specific link (representing a decision) and access additional information about the decision (Figure 19). This drill-down allows us to see all the calculations produced when we evaluated the cost (and feasibility) of covering a demand with a particular aircraft. For example, if we have to stop at intermediate airbases for refuelling, we can list these airbases.

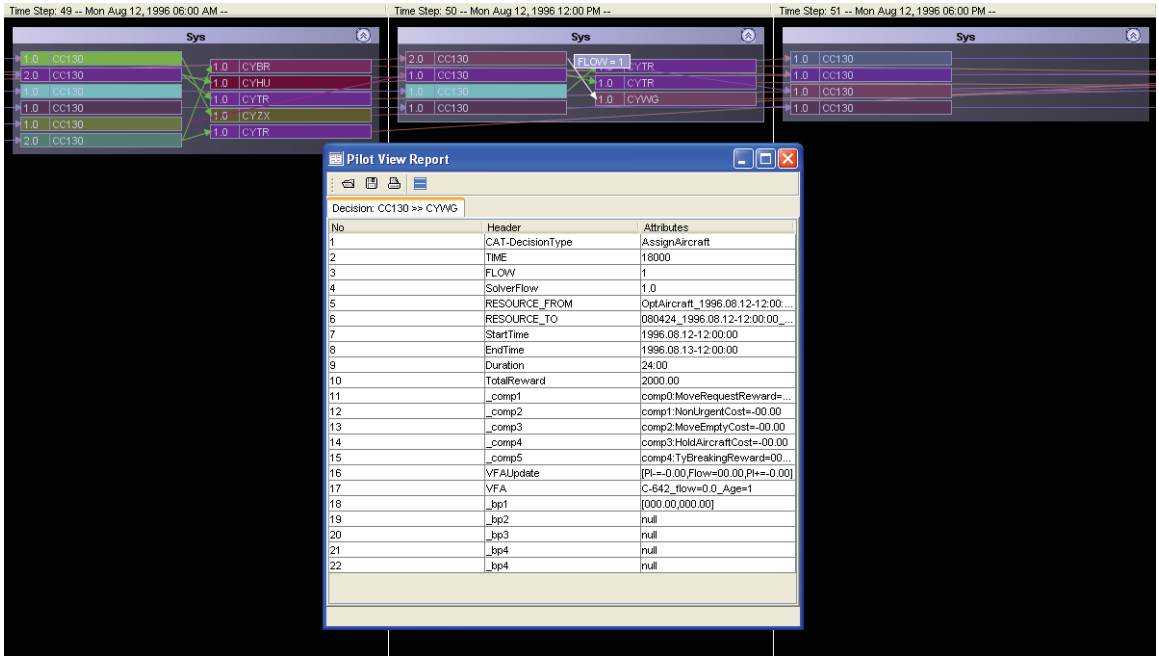

Figure 19: Drill down capability for individual decisions

6.2 Experiments on a deterministic dataset

Our first set of experiments assumed that all demands are known in advance. We started our experiments by testing the effect of using different planning horizons within an approximate dynamic programming algorithm. For these runs, when we made a decision at time *t*, we considered only aircraft and demands that were actionable within a specified planning horizon.

Figure 20 shows the percent of orders that were covered for planning horizons of 0, 12 and 24 hours, over the iterations of the approximate dynamic programming algorithm. The results show that the approximate dynamic programming algorithm produces a significant improvement in coverage if the planning horizon is set to zero, but achieving almost 99 percent coverage. When the planning horizon is lengthened to 6 or 12 hours, we obtain virtually 100 percent coverage even without the value functions.

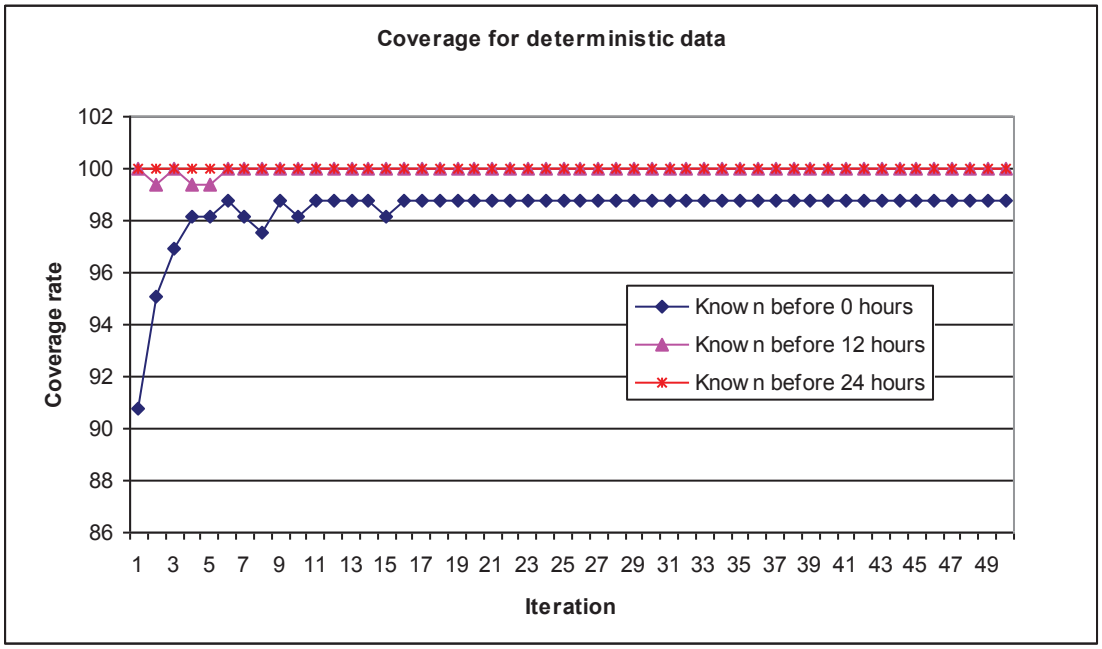

Figure 20: Effect of planning horizon and approximate dynamic programming for a deterministic dataset

A careful analysis of the data and these initial results showed us that the problem is extremely simple. All the aircraft are identical. If all the orders can be covered, and there is no differentiation between the aircraft, then looking into the future offers almost no value. The travel times are quite short. Most movements can be made within 6 hours, and virtually all can be made within 12 hours.

The most difficult problem we faced was the classical challenge of sequencing orders within their time windows. Typically, in a stochastic, dynamic problem, we are trying to serve as many orders as we can as quickly as possible. If we cannot serve all orders with available aircraft, we determine which ones to solve using a combination of assignment costs, the value of the order (determined by its priority) and the downstream value function. It is the value function (which gives the value of the aircraft after the order is completed) that helps us with the sequencing (for example, that we can serve a short demand now and still cover another demand later).
The execution times for the optimizing-simulator are not affected by the presence of wide time windows (by contrast, wide time windows can produce a dramatic increase in the execution times for column-generation methods). We can obtain very high quality solutions with tight time windows, or very wide time windows. When there is a mixture of tight and wide time windows, we will notice a modest degradation in solution quality (run times are the same under all scenarios).

6.3 Experiments with random demands

The next set of experiments introduced uncertainty in the generic loads. We assumed that each generic load was first learned at time τ^{pb} (the prebook time) before the demand had to be served. That is, if t is the *knowable time* (the time at which we learn about the demand), then the *actionable time* (the earliest time at which the demand can be served) is $t + \tau^{ph}$. For simplicity, we did not randomize τ^{pb}, although this would be quite easy to do. In addition, we randomized the loads themselves by taking the original set of generic loads, replicating each one five times, and then choosing a load from this set with probability 0.2.

Figure 21 shows the objective functions for prebook times of 0, 2 and 6 hours. Each run was performed with and without approximate dynamic programming. The value functions produce a noticeable improvement if the prebook time is 0, but the improvement is negligible when the prebook time is 2 or 6 hours.

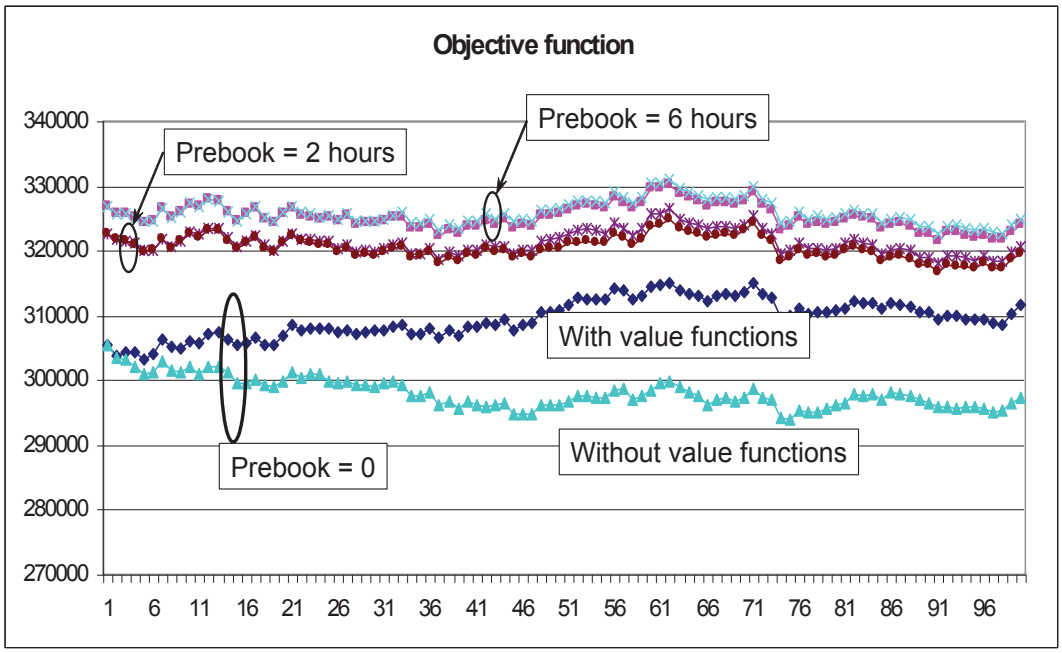

Figure 21: Objective function for prebook times of 0, 2 and 6 hours, with and without approximate dynamic programming

Figure 22 shows the percent of demands that were covered for the same runs. Again, we see a large improvement when we use approximate dynamic programming and a small (but noticeable) improvement when the prebook time is 2 hours. When we have six hours notice, coverage with and without the approximate dynamic programming is virtually 100 percent.

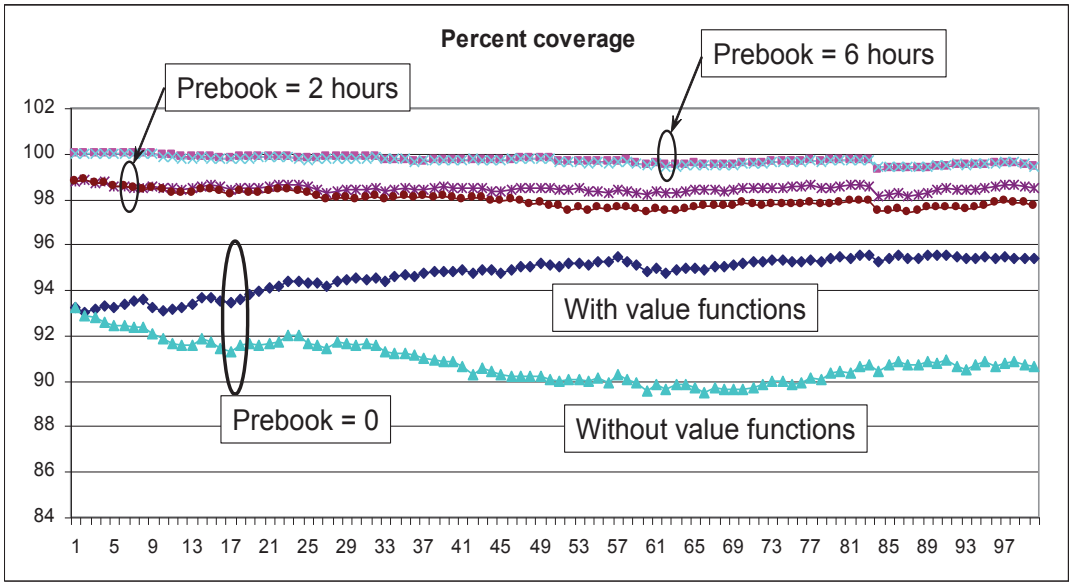

Figure 22: Demand coverage (as a percent of the total available) for prebook times of 0, 2 and 6, with and without approximate dynamic programming

These runs show that the performance of the system improves significantly as the prebook time ranges from 0, 2 and 6 hours. The value of the additional prebook time, however, is reduced significantly when we use value functions estimated using approximate dynamic programming. Without ADP, the improvement in the coverage as the prebook time increases from 0 to 2 hours appears to rise from just over 90 percent to 96 percent. With approximate dynamic programming, the coverage increases from just under 96 percent to 98 percent.

Figure 23 gives the difference between the coverage with and without approximate dynamic programming. We note that each run (with and without value functions) was performed with the exact same set of demand realizations (we use a different random sample of demands at each iteration, but use the exact same sample for each run).

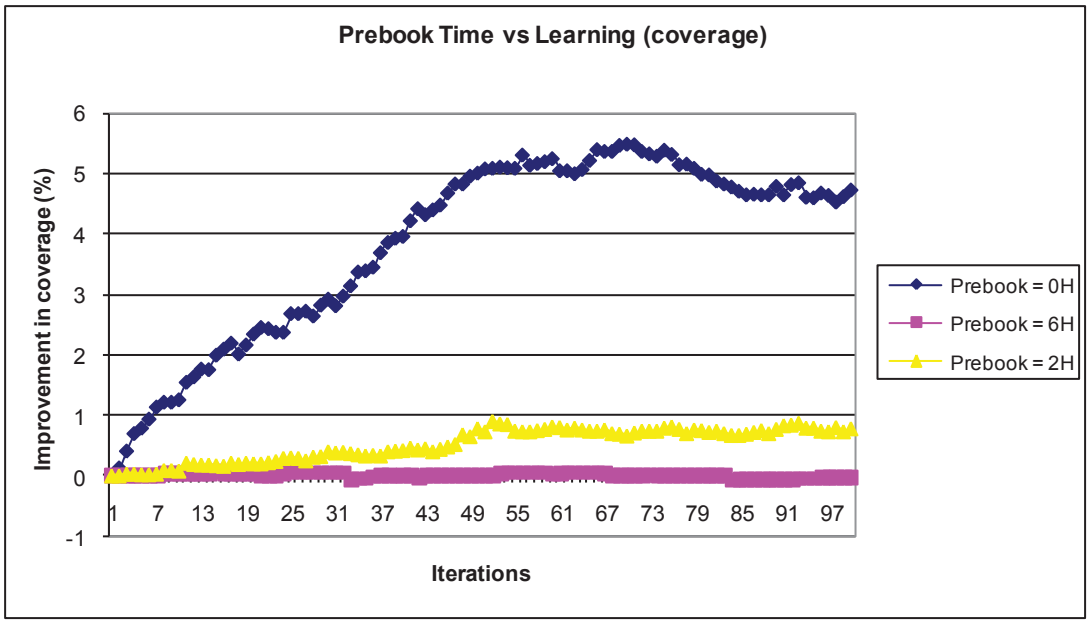

Figure 23: Improvement in demand coverage as a result of adaptive learning, for three different prebook times

6.4 Experiments with random aircraft failures

We next introduced the behaviour that aircraft may breakdown at the end of each segment. The first set of results is shown in Figure 24 which reports on the following runs:

a) Aircraft which break down and no adaptive learning (value functions are set to zero).

b) Aircraft which break down, but with adaptive learning.

c) Aircraft do not break down, and no adaptive learning adaptive learning (this number does not change with the iterations – there is no random sampling of breakdowns, and no learning).

d) Aircraft do not break down, but with adaptive learning.

All runs are shown in Figure 24 smoothed and unsmoothed (a smoothed curve is created by approximating a function that attempts to capture important patterns in the data),

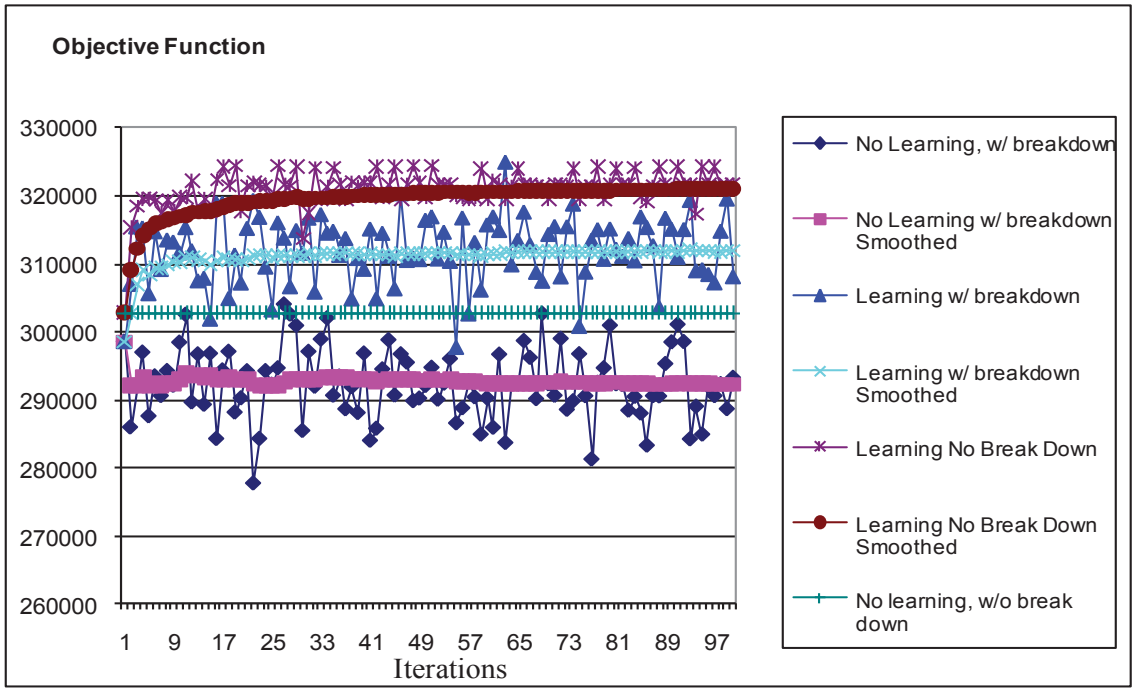

Figure 24: Objective function for a) breakdowns with no learning, b) breakdowns with learning, c) learning with no breakdowns. All runs are shown smoothed and unsmoothed.

These runs assumed demands were deterministic, but generic demands become known with no advance warning. The results clearly show that the solution quality degrades in the presence of equipment breakdowns, whether or not there is adaptive learning.

Figure 25 shows the reduction in profits as a result of introducing breakdowns. This change is shown with and without adaptive learning. The results show that the adaptive learning reduces the drop in profits due to equipment failures by approximately 15 percent (considering only the last 50 iterations).

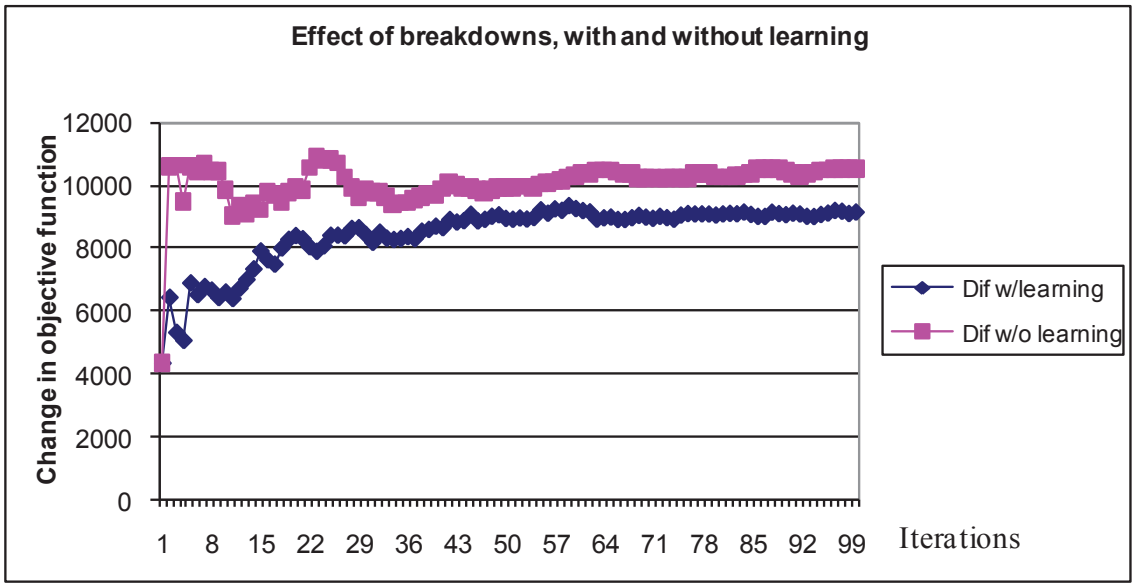

Figure 25: Reduction in profits due to breakdowns, with and without adaptive learning

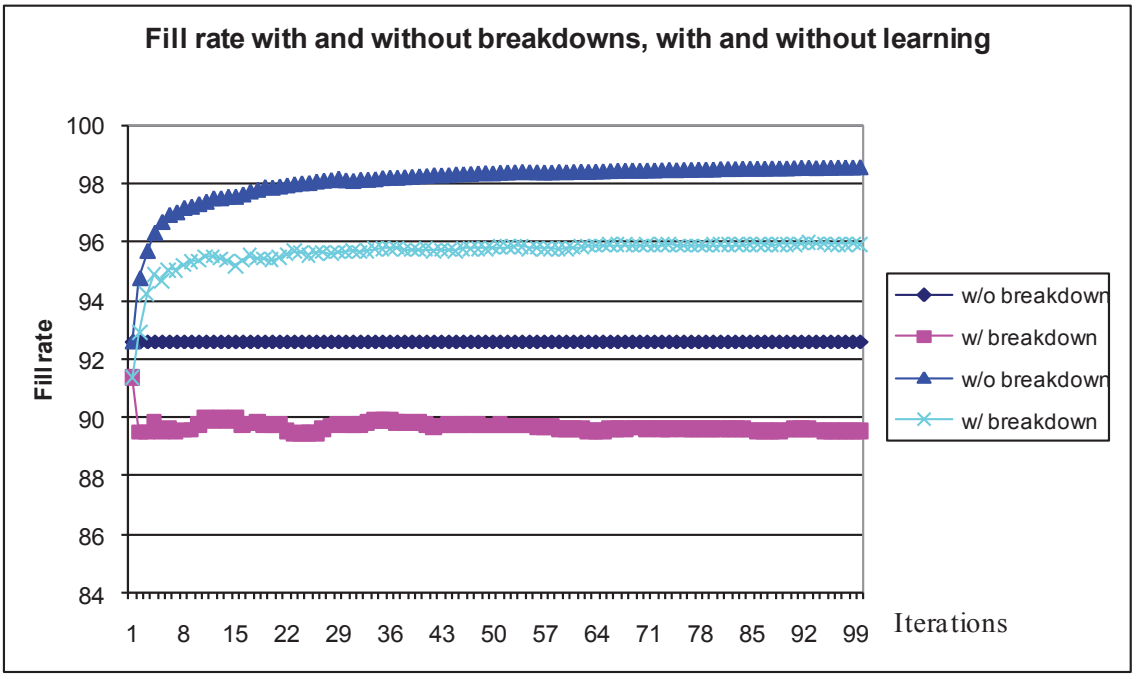

Figure 26: Order fill rate, with and without breakdowns, run with and without adaptive learning

Similar graphs are shown in Figure 26 and Figure 27 for the fill rate. Again, we get a measure of the effect of aircraft failures, and see that adaptive learning reduces this effect by approximately 15 percent.

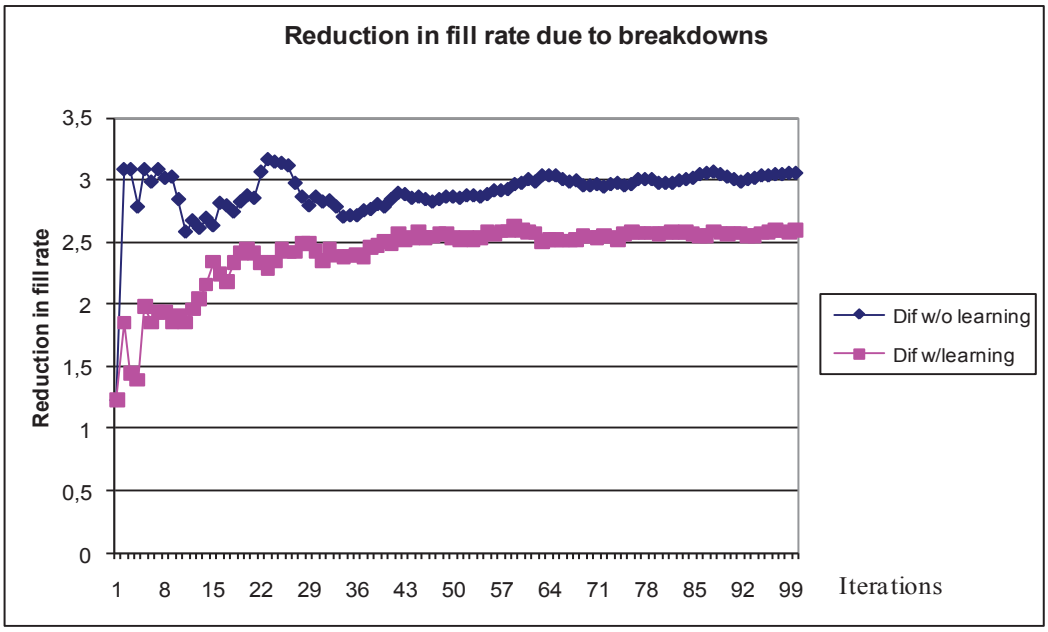

Figure 27: Reduction in fill rate due to breakdowns, measured with and without adaptive learning

Figure 28 shows the objective function for all combinations: random and deterministic demands, with and without aircraft failures, and with and without adaptive learning (eight runs in total). The runs are listed in the legend in the same order as the final objective function. These runs produce the following expected results:

- The best results are obtained with deterministic demands, no breakdowns and with adaptive learning.
- The worst results are obtained with random demands, aircraft failures and no learning.
- Randomness in demands or aircraft failures produces worse results than without this source of randomness (and all else held equal).
- Learning always outperforms no learning (with all else held equal).

Other results are not obvious, and probably depend on the specific dataset. For our experiments:

- Randomness in demands has a more negative impact than aircraft failures, with or without adaptive learning.
- Randomness in demands with adaptive learning outperforms deterministic demands and no learning, when aircraft breakdowns are allowed (this result is especially surprising).

These experiments are primarily designed to demonstrate the capability of the optimizing-simulator. In terms of validating the methodology, the important result is that the results are all intuitively reasonable. Specifically:

- Increasing the level of noise (random demands, aircraft breakdowns or both) reduces solution quality.
- Adaptive learning improves solution quality.
- Adaptive learning reduces the cost of uncertainty (eg. by measuring the effect of aircraft failures).

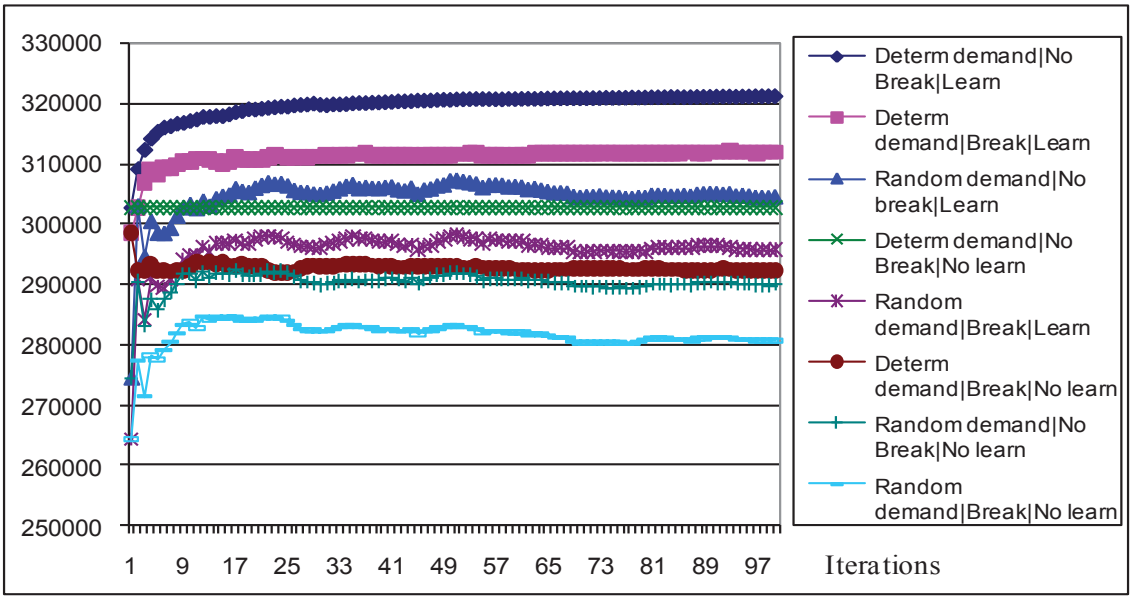

Figure 28: Objective function for all combinations: random and deterministic demands, with and without aircraft failures, and with and without adaptive learning

7 Conclusion

We showed in this report how the airlift problem can be modeled as a complex resource allocation problem, covering both the representation of the resources themselves and modeling the information. We also presented in this study a modeling Framework that is based on an optimizing-simulator to solve the airlift problem. The optimizing-simulator is an analysis technology that focuses on the modeling of complex resource allocation problems, using an explicit model of the information available to make a decision. We presented a series of models that illustrates how we model the information content of a decision, and the decision making technology. Using the specific context of military airlift, we presented a series of models of the information available to make a decision. We showed how this modeling framework can be used to represent the Canadian military airlift problem. We should mention that the dataset used to solve the problem have been prepared for an optimization model, and as such lacked the richness that we typically find in real applications.

The goal of this study is to compare, for the military airlift problem, the performance and capabilities of the optimizing-simulator technology (handling uncertainties) against that of classical deterministic optimization for the military airlift problem. For comparison, we also provide a brief comparison to classical simulation. The optimizing-simulator, rather than fostering a competition between alternative analysis technologies, actually combines the best features of simulation and optimization. In the process, it combines cost-based and rule-based decision technologies, providing an analysis technology that handles the high-dimensionality of problems that typically arise in transportation, as well as the complex behaviours that may be difficult to capture using a cost model, but which can be expressed as patterns of behaviour.

This report has shown how to model information flow and the airlift complex problem dynamics using a simple, intuitive vocabulary. We demonstrated this modeling framework in the context of the Canadian military airlift problem. We presented a series of computational experiments that evaluate the effect of random demands and equipment failures on system performance. The experiments clearly demonstrate the capability of the optimizing-simulator. In terms of validating the methodology, the important result is that the results are all intuitively reasonable. We observed that increasing the level of uncertainty (random demands, aircraft breakdowns or both) reduces solution quality. Also, the adaptive learning approach found to improve solution quality as well as reduce the cost of uncertainty.

A series of experiments were run to test the capabilities of the modeling and algorithmic framework. We demonstrated the quality of the solution achieved using the approximate dynamic programming algorithm. With no uncertainty, and with a modest level of advance information, a simple simulation provides very high quality solutions. We then assumed that generic demands became known over the simulation, where we varied the time between when a demand became known and when it had to be served (the prebook time). We were able to quantify the value of knowing orders in advance (we assumed 0, 2 and 6 hours), and we also showed that the value of advance information is reduced if we use adaptive learning logic.

We reported in this report the results comparing the objective function produced by the optimizing-simulator to optimal solutions for deterministic problems, and the results of rolling-horizon experiments for problems which exhibit uncertainty. The results of our experiments show that providing more information reduces costs and increases throughput.

Our experiments on the CF airlift problem clearly demonstrate the capability of the optimizing simulator. In terms of validating the methodology, the important result is that the results are intuitively reasonable. Specifically, we observed that increasing the level of noise (random demands, aircraft breakdowns or both) reduces solution quality. Also, adaptive learning was found to improve solution quality as well as reduce the cost of uncertainty.

References

[1] Dantzig, G. & Ferguson, A. (1956), 'The allocation of aircrafts to routes: An example of linear programming under uncertain demand', Management Science 3, 45–73.

[2] Wu Tongqiang, W.B. Powell and A. Whisman, (2008),"The Optimizing-Simulator: An Illustration using the Military Airlift Problem," ACM Transactions on Modeling and Simulation, Vol. 19, No. 3, Issue 14, pp. 1-31.

[3] Wing, V., Rice, R. E., Sherwood, R. & Rosenthal, R. E. (1991), Determining the optimal mobility mix, Technical report, Force Design Division, The Pentagon, Washington D.C.

[4] Yost, K. A. (1994), The thruput strategic airlift flow optimization model, Technical report, Air Force Studies and Analyses Agency, The Pentagon, Washington D.C.

[5] Rosenthal, R., Morton, D., Baker, S., Lim, T., Fuller, D., Goggins, D., Toy, A., Turker, Y., Horton, D. & Briand, D. (1997), 'Application and extension of the Thruput II optimization model for airlift mobility', Military Operations Research 3(2), 55–74.

[6] Killingsworth, P. & Melody, L. J. (1997), Should C17s be deployed as theater assets?: An application of the CONOP air mobility model, Technical report rand/db-171-af/osd, Rand Corporation. 2 Midler, J. L. &Wollmer, R. D. (1969), 'Stochastic programming models for airlift operations', Naval Research Logistics Quarterly 16, 315–330.

[7] Baker, S., Morton, D., Rosenthal, R. & Williams, L. (2002), 'Optimizing military airlift', Operations Research 50(4), 582–602.

[8] Crino, J. R., Moore, J. T., Barnes, J. W. & Nanry, W. P. (2002), 'Solving the theatre distribution vehicle routing and scheduling problem using group theoretic tabu search', Submitted to Mathematical and Computer Modelling 0(0), 17

[9] Burke, J. F. J., Love, R. J. & Macal, C. M. (2002), 'Modelling force deployments from army installations using the transportation system capability (TRANSCAP) model: A standardized approach', Submitted to Mathematical and Computer Modelling 0(0), 00. 3 24.

[10] Midler, J. L. & Wollmer, R. D. (1969), 'Stochastic programming models for airlift operations', Naval Research Logistics Quarterly 16, 315–330

[11] Goggins, D. A. (1995), Stochastic modeling for airlift mobility, Master's thesis, Naval Postgraduate School, Monterey, CA.

[12] Niemi, A. (2000), Stochastic modeling for the NPS/RAND Mobility Optimization Model, Department of Industrial Engineering, University of Wisconsin-Madison, Available: http://ie.engr.wisc.edu/robinson/Niemi.htm.

[13] Morton, D. P., Salmeron, J. & Wood, R. K. (2002), 'A stochastic program for optimizing military sealift subject to attack', Stochastic Programming e-print series. http://www.speps.info.

[14] Yost, K. A. & Washburn, A. R. (2000), 'The LP/POMDP marriage: Optimization with imperfect information', Naval Research Logistics 47(8), 607–619.

[15] Bertsekas, D. & Tsitsiklis, J. (1996), Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[16] Sutton, R. & Barto, A. (1998), Reinforcement Learning, The MIT Press, Cambridge, Massachusetts.

[17] Si, J., Barto, A. G., Powell, W. B. & D. Wunsch II, e. (2004), Handbook of Learning and Approximate Dynamic Programming, IEEE Press, New York.

[18] Si J., A. Barto, W.B. Powell and D. Wunsch (eds.), (2004) Learning and Approximate Dynamic Programming: Scaling up to the Real World, John-Wiley and Sons, New York.

[19] Powell, W. B. & Van Roy, B. (2004), Approximate dynamic programming for high dimensional resource allocation problems, in J. Si, A. G. Barto, W. B. Powell & D. Wunsch II, eds, 'Handbook of Learning and Approximate Dynamic Programming', IEEE Press, New York.

[20] Godfrey, G. and W.B. Powell, "An Adaptive Dynamic Programming Algorithm for Single-Period Fleet Management Problems I: Single Period Travel Times," Transportation Science, Vol. 36, No. 1, pp. 21-39 (2002).

[21] Godfrey, G. and W.B. Powell, (2002),"An Adaptive Dynamic Programming Algorithm for Single-Period Fleet Management Problems II: Multiperiod Travel Times," Transportation Science, Vol. 36, No. 1, pp. 40-54.

[22] Powell, W. B., Shapiro, J. A. & Simao, H. P. (2002), 'An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem', Transportation Science 36(2), 231–249.

[23] Spivey, M. & Powell, W. B. (2004), 'The dynamic assignment problem', Transportation Science 38(4), 399–419.

[24] Powell, W. B. & Topaloglu, H. (2004), Fleet management, in S. Wallace & W. Ziemba, eds, 'Applications of Stochastic Programming', Math Programming Society - SIAM Series in Optimization, Philadelphia.

[25] Wu, T. T., Powell, W. B. & Whisman, A. (2003), The optimizing simulator: An intelligent analysis tool for the military airlift problem, Technical report, Princeton University, Department of Operations Research and Financial Engineering.

[26] Powell, W. B., Wu, T. T. & Whisman, A. (2004), 'Using low dimensional patterns in optimizing simulators: An illustration for the airlift mobility problem', Mathematical and Computer Modeling 29, 657–665.

[27] Powell, W.B., J. Shapiro and H.P. Simao, (2001) "A Representational Paradigm for Dynamic Resource Transformation Problems," Annals of Operations Research on Modeling (C. Coullard, R. Fourer, and J. H. Owen, eds), Vol. 104, pp. 231-279.

[28] Powell, W.B., T. Wu, H. P. Simao and A. Whisman, (2004) "Using Low Dimensional Patterns in Optimizing Simulators: An Illustration for the Military Airlift Problem," Mathematical and Computer Modeling 29, pp. 657-675.

[29] Topaloglu, H. and W.B. Powell, (2005) "A Distributed Decision-Making Structure for Dynamic Resource Allocation with Nonlinear Functional Approximations," Operations Research, Vol. 53, No. 2, pp. 281-297.

[30] Powell, W.B., A. Ruszczynski and H. Topaloglu, (2004) "Learning Algorithms for Separable Approximations of Stochastic Optimization Problems," Mathematics of Operations Research, Vol. 29, No. 4, pp. 814-836.

[31] Powell, W.B. (2007), *Approximate Dynamic Programming: Solving the curses of dimensionality, John Wiley and Sons,* pp. 488.

List of symbols/abbreviations/acronyms/initialisms

AF	Air Force
CF	Canadian Forces
DND	Department of National Defence
DRDC	Defence Research & Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
R&D	Research & Development
DRTP	Dynamic Resource Transformation Problem

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC JUNE 2010

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Optimal Decisions for Canadian Military Airlift Problem

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Boukhtouta, A., Berger, J., Powell, W.B., Bouziane-Ayari, B.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
September 2010	72	31

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
13du	

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Valcartier TR 2010-517	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

A modeling and algorithmic framework for analyzing and simulating the military airlift is presented in this report. Central to the framework is the modeling of the flow of information and decisions. A series of models demonstrating how different levels of information can be represented when making a decision are presented. These information sets are simulated to demonstrate their impact on costs and throughput. A detailed comparison of classical optimization and the current framework is presented. Using historical data for the airlifts conducted by the Canadian Air Force, a series of simulations were conducted to test the effect of uncertainty in customer demands as well as aircraft failures. It is demonstrated in this report that this effect is reduced when adaptive learning is used in the simulation process. There are a number of sources of randomness that arise in military airlift operations: random demands, aircraft failures, weather delays, etc. The cost of uncertainty can be difficult to estimate because it is difficult to give the mathematical expression of this cost. However, it is easy to overestimate this cost if we use simplistic decision rules. The military airlift problem, considered in this report, is modeled as a dynamic program, and solved using approximate dynamic programming. The experiments show that even approximate solutions produce decisions that substantially reduce the effect of uncertainty.

Une approche de modélisation et de résolution pour l'analyse et la simulation du problème du transport aérien militaire est présentée dans ce rapport. La modélisation des flots d'informations et de décisions en constitue l'élément central. Nous présentons plusieurs modèles pour montrer comment l'information est traitée et présentée durant le processus de prise de décision. Plusieurs scénarios sont simulés pour déterminer l'impact de l'information sur les coûts ainsi que sur la solution. Nous faisons une comparaison détaillée entre l'approche d'optimisation classique et celle qui est proposée dans ce rapport qui est le simulateur d'optimisation. En utilisant des données historiques des opérations aériennes des Forces Canadiennes, une série de simulations a été effectuées pour mesurer l'impact des incertitudes liées aux requêtes de dernière minute et aux bris mécaniques des avions. Nous démontrons que cet impact est réduit lorsque nous utilisons le concept d'apprentissage adaptatif dans le processus de simulation. Il y a plusieurs sources d'incertitude liées au problème du transport militaire aérien : requêtes imprévues, bris d'avions, retards dus à la météo, etc. L'estimation du coût d'incertitude est difficile à obtenir puisque ce dernier est difficile à exprimer mathématiquement. Par ailleurs, ce coût est facile à surestimer quand on utilise des règles de décisions simplistes. Le problème du transport militaire aérien abordé dans ce rapport est modélisé comme un programme dynamique qui est résolu via la programmation dynamique approximative. Les expériences numériques montrent que même les solutions approximatives donnent des décisions qui réduisent substantiellement l'effet d'incertitude.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

airlift problem; approximate dynamic programming; simulation; uncertainties

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca