# The Effect of Robust Decisions on the Cost of Uncertainty in Military Airlift Operations

Warren B. Powell, Princeton University
Belgacem Bouzaiene-Ayari, Princeton University
Jean Berger, DRDC-Valcartier
Abdeslem Boukhtouta, DRDC-Valcartier
Abraham P. George, Princeton University

There are a number of sources of randomness that arise in military airlift operations. However, the cost of uncertainty can be difficult to estimate, and is easy to overestimate if we use simplistic decision rules. Using data from Canadian military airlift operations, we study the effect of uncertainty in customer demands as well as aircraft failures, on the overall cost. The system is first analyzed using the types of myopic decision rules widely used in the research literature. The performance of the myopic policy is then compared to the results obtained using robust decisions which account for the uncertainty of future events. These are obtained by modeling the problem as a dynamic program, and solving Bellman's equations using approximate dynamic programming. The experiments show that even approximate solutions to Bellman's equations produce decisions that reduce the cost of uncertainty.

Categories and Subject Descriptors: I.6.1 [**Simulation and Modeling**]: Simulation theory; I.6.3 [**Simulation and Modeling**]: Applications; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

General Terms: Algorithms,theory,experimentation

Additional Key Words and Phrases: Approximate dynamic programming,robust control,military logistics

**ACM Reference Format:**
Powell, W.B., Bouzaiene-Ayari, B., Berger, J., Boukhtouta, A., George, A. P. 2011. The Effect of Robust Decisions on the Cost of Uncertainty in Military Airlift Operations. ACM Trans. Model. Comput. Simul. 9, 4, Article 39 (March 2010), 19 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

## 1. INTRODUCTION

The problem of military airlift involves managing a fleet of aircraft to serve customer demands to move passengers or freight. Each aircraft is characterized by a vector of 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 measures of operability. Each of the demands (known in the military as "requirements") is also described using features such as the type of demand, their origin, the type of aircraft that is required to serve the demand, their priority, and the pickup

This work is supported by the DRDC-Valcartier, Defense research of Canada, and the Air Force Office of Scientific Research, contract FA9550-08-1-0195.
Author's addresses: W. B. Powell, B. Bouzaiene-Ayari and A. P. George, Department of Operations Research and Financial Engineering, Princeton University; J. Berger and A. Boukhtouta, DRDC-Valcartier, Quebec, Canada.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

and delivery time windows. The aircraft have to undertake tasks such as picking up and moving the passengers and goods, and moving empty to another location to serve demands arising there. The demands are made up of a mixture of scheduled requests, which are known in advance, and dynamic requests, which arrive over time with varying degrees of advance notice. Other forms of uncertainty, such as the random failure of equipment, can also occur.

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 ([Wing et al. 1991]). [Yost 1994] describes the development of THRUPUT which is a static airlift model which does not capture the timing of events. THRUPUT and the Mobility Optimization Model were combined to produce THRUPUT II ([Rosenthal et al. 1997]) which modeled airlift operations in the context of a time-dependent model. A similar model produced by RAND called CONOP (CONcept of OPerations) is described in [Killingsworth and Melody 1997]. Both THRUPUT II and CONOP possess desirable features which were merged in a system called NRMO ([Baker et al. 2002]). These models could all be solved using mathematical programming packages. [Crino et al. 2004] addresses the problem of routing and scheduling vehicles in the context of theater operations. The resulting model was solved using group-theoretic tabu search.

Despite the attention given to math 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. 2004]). TRANSCAP is 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. These simulation models provide for a high level of realism, but they use simple myopic policies (such as finding the closest aircraft to serve a requirement) that do not produce robust solutions.

Since 1956 ([Dantzig and Ferguson 1956]) there has been interest in solving problems where decisions explicitly account for uncertainty in the future. Much of this work has evolved within the discipline of stochastic programming, which focuses on representing uncertainty within linear programs. [Mulvey et al. 1995] provides an introduction to the use of stochastic programming in a variety of applications, including a model called STORM which was designed to introduce uncertainty into the military airlift problem. [Midler and Wollmer 1969] accounts for uncertainty in demands, while [Goggins 1995] proposes an extension of THRUPUT II to handle uncertainty in aircraft reliability. [Niemi 2000] proposes a stochastic programming model which captures uncertainty in ground times. [Morton et al. 2003] introduced SSDM (Stochastic Sealift Deployment Model) to model sealift operations in the presence of possible attacks. [List et al. 2003] illustrates the use of stochastic programming for fleet sizing, which can be thought of as a design question, while our focus is on how to control a fleet under uncertainty.

Our analysis approach is based on the field of approximate dynamic programming (ADP) ([Bertsekas and Tsitsiklis 1996], [Sutton and Barto 1998], [Powell 2007]) 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 mathematical programming (which is needed to handle the high dimensional problems that arise in transportation) with the flexibility of simulation (see, for example, [Powell and Van Roy 2004]) using the

framework of dynamic programming. ADP is a simulation-based optimization algorithm. However, the resulting policies are not optimal, so the research challenge is to show that we can obtain high quality solutions that are robust under different scenarios.

ADP produces decisions that are robust, which is to say that they work well under different potential outcomes in the future. A model might send five aircraft to handle work that could be covered by three or four aircraft in anticipation of possible aircraft failures. It might also position extra aircraft at locations which can handle potential failures in nearby locations. In our work, we use ADP to produce robust decisions, and we show that the decisions produced using ADP are, in fact, less sensitive to uncertainty.

This paper makes the following contributions. 1) It provides an optimization-based model of airlift operations that enables as much detail as existing simulation models (comparable to that used in [Wu et al. 2009] and existing airlift simulators). 2) It demonstrates, using data from Canadian airlift operations, that ADP will produce robust solutions by showing that decisions perform better over a range of scenarios than decisions produced using myopic or deterministic models. 3) We quantify the value of advance information, and show that the value of advance information is reduced when we use robust decisions. 4) We quantify the impact of random demands and random equipment failures with and without using robust decisions. The techniques we use in this paper have been used elsewhere (notably [Topaloglu and Powell 2006]), but this is the first time that we have explicitly demonstrated robust behavior using ADP. In addition, we show that robust behavior changes in a significant way the cost of uncertainty in the context of a military airlift problem.

Section 2 provides a model of the Canadian military airlift problem using the notation and vocabulary of dynamic resource management. In section 3, we describe our strategy for obtaining robust policies using an ADP approach where we account for the effect of decisions now on the future operation. Section 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 that we use. We provide concluding remarks in section 5.

## 2. PROBLEM FORMULATION

In this section, we present a model of the Canadian airlift problem. We model the airlift problem as a two-layer, heterogeneous resource allocation problems, where "aircraft" and "demands" represent the two resource layers. We represent the characteristics of an aircraft using

$a$ = Vector of attributes describing an aircraft, e.g., type, location, time of availability.

$\mathcal{A}$ = Set of all possible aircraft attributes.

The *time of availability* of an aircraft denotes the actionable time, which is the earliest point in time when the plane can be assigned to a new demand or moved empty to another location. For example, a plane that is enroute to a destination will generally not be actionable until it arrives.

Similarly we use $b$ to represent the vector of attributes (such as origin, departure time window and knowable time) of a demand and $\mathcal{B}$ the set of all possible attributes of demands. The *knowable time* of a demand represents the time at which it becomes known (often referred to as the "call-in" time). In the Canadian airlift problem, some demands are known in advance, while others are "dynamic" and only become known as the system evolves.

Instead of having hard constraints on the time windows for serving demands, we impose a penalty for moving the demand after the end of the time window. We assume that the beginning of the departure time window is the earliest time that a demand is available to be moved.

Letting "A" denote aircraft, the state of the aircraft is described using

$$\begin{aligned} R_{ta}^A &= \text{The number of aircraft with attribute } a \text{ at time } t \text{ just before we make a decision,} \\ R_t^A &= \left(R_{ta}^A\right)_{a\in\mathcal{A}}. \end{aligned}$$

Similarly, letting "D" represent demands, $R_{tb}^D$ denotes the number of demands with attribute $b$ at time $t$ just before we make a decision and $R_t^D$ the vector $\left(R_{tb}^D\right)_{b\in\mathcal{B}}$. The state of the system can be written as $R_t = \left(R_t^A, R_t^D\right)$.

We now define decisions as actions that can change the state of the system. For the Canadian airlift problem, the only decisions for aircraft are how to move a demand, whether to move aircraft empty to a location, or whether to do nothing. Letting $\mathcal{D}$ denote the different types of decisions, we use the following notation to represent decisions:

$$\begin{aligned} x_{tad} &= \text{Number of resources with attribute } a \text{ acted on with a decision of type } d \in \mathcal{D} \text{ using the information available at time } t, \\ x_t &= \text{Vector of all types of decisions acting on all the aircraft at time } t, \\ &= \left(x_{tad}\right)_{a\in\mathcal{A},d\in\mathcal{D}}. \end{aligned}$$

For our model, $\mathcal{D}$ includes decisions to reposition empty to a location, and decisions to move a type of demand.

Next, we introduce the notion of a policy $\pi$ which is a rule for choosing a decision given the information available. We use $\Pi$ to denote the set of all policies. Decisions are made using a ***decision function*** which we represent using

$$X_t^\pi(R_t) = \text{Function returning a feasible decision vector } x_t \text{ at time } t \text{ under a policy } \pi \text{ when we are in resource state } R_t.$$

We typically have a family of decision functions, where for $\pi \in \Pi$, $X_t^\pi$ is a particular decision function. Our challenge, stated formally below, is to choose the best decision function.

In addition to the decisions, exogenous information may also bring about changes in the resource attributes. This could involve new customer demands, changes to an existing customer demand or equipment failures. We let

$$W_t = \text{Exogenous information arriving between times } t-1 \text{ and } t.$$

We let $\omega$ be an instance of $(W_1, W_2, \ldots, W_T)$, where $\omega \in \Omega$, $\Omega$ is the set of all sample realizations of exogenous information, and $T$ is the number of time periods in our model. In a formal stochastic model, we would define a probability space $(\Omega, \mathcal{F}, \mathcal{P})$ where $(\mathcal{F}_t)_{t=1}^T$ is a set of filtrations on $\Omega$, and $\mathcal{P}$ is a probability measure on $(\Omega, \mathcal{F})$. We note that by construction, the decision function $X_t^\pi(R_t)$ is $\mathcal{F}_t$-measurable. We adopt the notational style that any variable indexed by $t$ is $\mathcal{F}_t$-measurable.

We use a transition function, $R^M$, to capture the evolution of the system (described in detail in Appendix A.1) over time as a result of the decisions and the exogenous information. We use the following transition function to represent this evolution,

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

We now define a ***contribution function*** that captures the benefit of activities such as moving a demand, as well as penalties and bonuses for specific behaviors. Table

Table I. Illustration of different elements of the contributions from assigning an aircraft to a demand.

| Contribution-category | "Cost"/"Bonus" ($) |
|---|---|
| Demand satisfaction | 18000 |
| Repositioning cost | -17000 |
| Appropriate aircraft type | 5000 |
| Requires modification | -3000 |
| Special maintenance at airbase | -1000 |
| Total "contribution" | 2000 |

I illustrates the different types of penalties and bonuses that are used to evaluate a decision. In practice, these numbers would reflect a combination of measured statistics (e.g. lateness in moving a requirement, use of an appropriate aircraft type, did the aircraft require modification) multiplied by coefficients that reflect the importance of each dimension to a planner, which is a subjective judgment. These will typically be a mixture of hard-dollar components, for example, the cost of moving to a particular location, and soft-dollar components, such as a penalty for serving a demand late, or a bonus for using a particular type of aircraft to move passengers. We define

$$\begin{aligned} c_{tad} &= \text{The contribution from acting on a resource (aircraft) with attribute } a \\ &\quad \text{using a decision of type } d \in \mathcal{D} \text{ at time } t, \\ C_t(R_t, x_t) &= \text{The total contribution at time } t \text{ given state } R_t \text{ and action } x_t \\ &= \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} c_{tad} x_{tad}. \end{aligned}$$

We note that we are using the standard language of dynamic programming when writing the total contribution as a function of $R_t$. In our model, the contribution is purely a function of $x_t$, which itself is a function of $R_t$. However, in a more realistic model, the cost of assigning an aircraft might depend on the congestion at an airbase or the queue of aircraft waiting for refueling or maintenance.

The Canadian airlift problem involves finding the best decision function (policy) $X_t^\pi(R_t)$ that solves

$$\max_{\pi \in \Pi} \mathbb{E} \left\{ \sum_{t=0}^{T} C\left(R_t, X_t^\pi(R_t)\right) \right\}. \tag{2}$$

We could solve this problem using a myopic policy by stepping through time, using a sequence of assignment problems such as is illustrated in Fig. 1. At time $t$, we would consider aircraft and demands that are known now (at time $t$) but may be actionable now or at some time $t' > t$ in the future. These problems are typically quite small and can be solved very quickly. Most of the computer time is spent in generating the network and computing the costs. For airlift problems, simply determining the cost of an option can be expensive (this may require running a small simulation to determine if the intermediate air bases have the capacity to handle the aircraft).

The value of using dynamic programming as a framework for approximately solving (2) is not just to obtain a better solution, but to obtain a more realistic solution. A myopic policy might ignore, for example, the inability of a particular location to solve maintenance problems for specific types of aircraft. It would also ignore the fact that we might need four aircraft at a location when we only anticipate needing three, simply because one of the aircraft may need unscheduled maintenance. We may also need additional aircraft to handle delays in inbound aircraft due to weather. These decisions have to reflect the actual state of the system, since unexpected failures and delays will require changes to the plan. In this paper, we are particularly interested in our

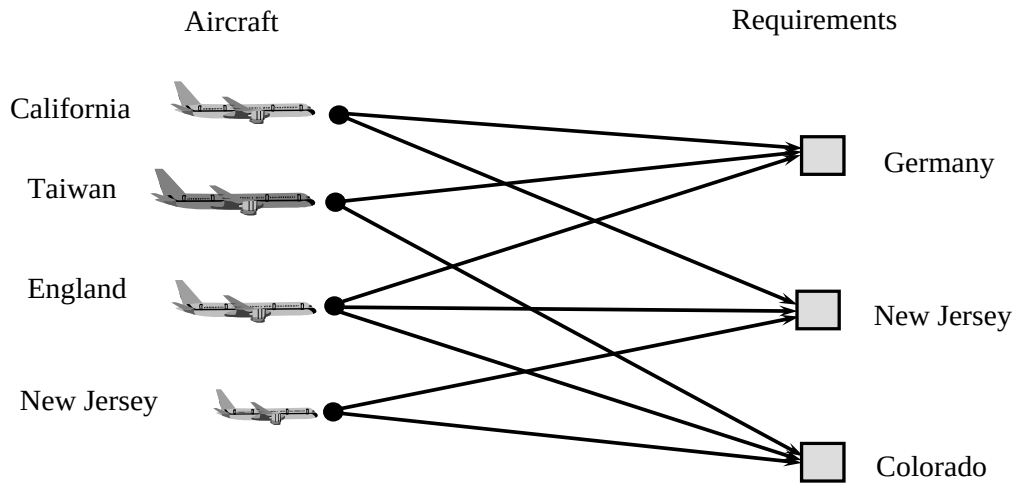


Fig. 1. An assignment problem considering multiple aircraft and demands.

ability to find robust policies, since these may have the effect of reducing the cost of uncertainty, which can impact the value of investments designed to reduce uncertainty.

## 3. SOLUTION STRATEGY

The most common strategies for solving equation (2) are either a) to use rule-based policies within a simulator or b) to formulate the entire problem deterministically as a single, large linear (or integer) program. [Morton et al. 2003] uses stochastic programming for a version of the sealift problem, limited to solve problems with a very small set of scenarios, and unable to model the movement of ships, congestion at ports, and the possibility that decisions within the model may actually influence random events (e.g. the behavior of the enemy). The Canadian airlift problem is an instance of a multistage stochastic integer program. The central thesis of this paper is that solving this stochastic optimization problem will produce robust strategies that are less sensitive to noise, a claim that is supported by our numerical experiments. The challenge is that the resulting optimization problem for finding optimal policies is computationally intractable. In this section, we show how to use the techniques of approximate dynamic programming to solve multistage linear programs. Our strategy is based on the concepts given in [Powell 2007].

In the remainder of this section, we describe the solution of this problem using ADP.

### 3.1. Approximate Dynamic Programming

We can solve our problem using dynamic programming where we let $V_t(R_t)$ be the value of being in state $R_t$. Bellman's optimality equation would then be written

$$V_t(R_t) = \max_{x_t \in \mathcal{X}_t} \left( C(R_t, x_t) + \mathbb{E}\{V_{t+1}(R^M(R_t, x_t, W_{t+1}))|R_t\} \right) \tag{3}$$

where the feasible region $\mathcal{X}_t$ satisfies non-negativity and integrality of the decision variables as well as conservation of flow. Here, $R^M(R_t, x_t, W_{t+1})$ gives the state $R_{t+1}$ as a function of the current state $R_t$, the decisions $x_t$ and the new information $W_{t+1}$. We note that our constraint set only captures constraints that directly act on our choice of $x_t$ such as conservation of flow or limits on which aircraft can handle a particular

requirement. In a more traditional math programming formulation, we would also have to include in the constraint set the equations that govern the evolution of the system over time (these are now captured in the transition function in equation (1)).

Using standard techniques, Bellman's equation can be solved to optimality and the existence of optimal solutions is proved in works on traditional dynamic programming such as [Puterman 1994]. However, such techniques encounter the classic curse of dimensionality. We avoid this by breaking the transition function into two steps: the transition from the *pre-decision* state variable $R_t$ to the *post-decision* state $R_t^x$, where $R_t^x$ is the state of the system at time $t$, immediately after we have made a decision. We break the resource transition function $R^M(\cdot)$ into two functions which produce

$$R_t^x = R^{M,x}(R_t, x_t), \tag{4}$$

$$R_{t+1} = R^{M,W}(R_t^x, W_{t+1}). \tag{5}$$

Using the pre- and post-decision states allows us to break Bellman's equation into two steps, given by

$$V_t(R_t) = \max_{x_t \in \mathcal{X}_t} \left(C(R_t, x_t) + V_t^x(R_t^x)\right), \tag{6}$$

$$V_t^x(R_t^x) = \mathbb{E}\left\{V_{t+1}(R_{t+1})|R_t^x\right\}, \tag{7}$$

where $V_t(R_t)$ is the traditional value of being in (pre-decision) state $R_t$, while $V_t^x(R_t^x)$ is the value of being in post-decision state $R_t^x$. Note that, when considered separately, equation (6) is completely deterministic, while equation (7) involves nothing more than an expectation (which is computationally intractable for most applications, including ours).

We note that using a post-decision state is not an approximation (this idea is used in all decision trees). Calculating the value function around the post-decision state is equivalent to computing the expectation in (3) given the current state and action (which deterministically gives us the post-decision state). Once we have made a decision (by solving (6)), we step forward in time, at which point we learn if an aircraft has been delayed or if there has been an equipment failure. The marginal value of additional resources, computed after this information becomes known, is smoothed in the value function computed as a function of the post-decision state.

The next step is to replace the function $V_t^x(R_t^x)$ with an approximation. There are a number of strategies we can use. For our work, we used

$$\begin{aligned} V_t^x(R_t^x) &= \bar{V}_t(R_t^x) \\ &= \sum_{a \in \mathcal{A}} \bar{V}_{ta}(R_{ta}^x), \end{aligned}$$

where $\bar{V}_{ta}(R_{ta}^x)$ is piecewise linear and concave in $R_{ta}^x$, given by

$$\bar{V}_{ta}(R_{ta}^x) = \left(\sum_{r=0}^{\lfloor R_{ta}^x \rfloor} \bar{v}_{ta}(r) + (R_{ta}^x - \lfloor R_{ta}^x \rfloor)\bar{v}_{ta}(\lfloor R_{ta}^x \rfloor)\right), \tag{8}$$

where $\lfloor R \rfloor$ is the largest integer less than or equal to $R$. This function, illustrated in Fig. 2 is parameterized by the slopes $(\bar{v}_{ta}(r))$ for $r = 0, 1, \ldots, R^{max}$, where $R^{max}$ is an upper bound on the number of resources of a particular type, while $\lfloor R_{ta}^x \rfloor$ is the actual number of resources (rounded down) with attribute $a$ at time $t$. This approximation has worked well in a number of fleet management problems ([Powell and Godfrey 2002], [Powell and Topaloglu 2005], [Topaloglu and Powell 2006], [Wu et al. 2009]).

The approximation of the value function is done using an iterative procedure. We let $n$ denote the iteration counter obtained while following sample path $\omega^n$. Let $R_t^n$ be the

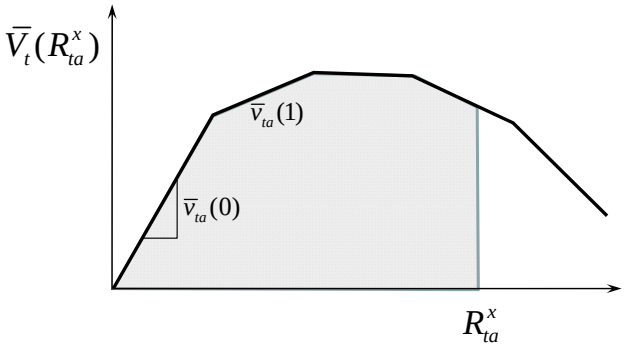


Fig. 2. Illustration of piecewise linear value function approximation.

state at time $t$ while following sample path $\omega^n$. At iteration $n$, we solve the decision problem given by

$$x_t^n(R_t^n) \;=\; \arg\max_{x_t \in \mathcal{X}_t^n} \left(C(R_t^n, x_t) + \bar{V}_t^{n-1}(R^{M,x}(R_t^n, x_t))\right). \tag{9}$$

For our application, equation (9) is an integer program (specifically, an integer multicommodity flow problem) which can be solved using a commercial solver. Let $\tilde{V}_t(R_t^n)$ be the objective function value obtained by solving (9). We can update our value function using estimates of the derivative of $\tilde{V}_t(R_t)$ with respect to $R_t$. If we compute this in a brute force manner, we would compute

$$\hat{v}_{ta}^n = \tilde{V}_t(R_t^n + e_{ta}) - \tilde{V}_t(R_t^n)$$

where $e_{ta}$ is a vector of 0's with a 1 in the element corresponding to $(t, a)$. In practice, we often use the dual variable of the linear programming relaxation of (9) associated with the flow conservation constraint, $\sum_{d\in\mathcal{D}} x_{tad} = R_{ta}$.

The last step of the process is to use the vector $\hat{v}_t^n$ to update the value function approximation $\bar{V}_{t-1}^{n-1}(R_{t-1}^{x,n})$ around the *previous* post-decision state $R_{t-1}^{x,n}$. Assume that $r = R_{t-1,a}^{x,n}$ is the number of cargo aircraft with attribute $a$ at $t-1$ after we made our previous decision. We would then update the slope corresponding to $r$ using

$$\bar{v}_{t-1,a}^n(r) = (1-\alpha_{n-1})\bar{v}_{t-1,a}^{n-1}(r) + \alpha_{n-1}\hat{v}_{ta}^n, \tag{10}$$

where $\alpha_{n-1}$ is a stepsize. In our work, we used the adaptive stepsize given in [George and Powell 2006]. After this update, we might find we have a concavity violation, which is to say that we no longer have $\bar{v}_{t-1,a}^n(r) \geq \bar{v}_{t-1,a}^n(r+1)$ for all $r$. This is easily fixed using any of the methods described in [Powell and Godfrey 2001], [Topaloglu and Powell 2003] or [Powell et al. 2004]. For ease of reference, we are going to refer to the entire process of updating the value function using

$$\bar{V}_{t-1}^n(R_{t-1}^x) = U^V(\bar{V}_{t-1}^{n-1}(R_{t-1}^x), R_{t-1}^{x,n}, \hat{v}_t^n).$$

Here, $U^V(\cdot)$ refers to any of a variety of methods for updating value functions given the new observation $\hat{v}_t^n$. Different methods are reviewed in [Lagoudakis and Parr 2003], [Menache et al. 2005], [Sutton and Barto 1998], [Bertsekas 2009], and [Szepesvari 2010], to name a few.

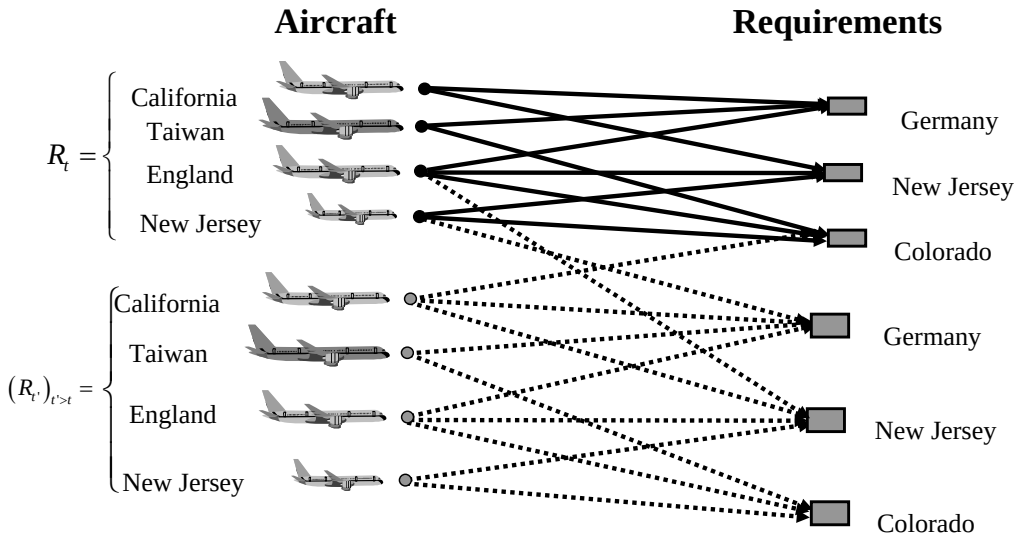


Fig. 3. Decision-problem, assigning aircraft to demands or to locations.

### 3.2. The decision problem

At any one point in time, aircraft can be assigned to demands, or moved to physical locations (see Fig. 3). A movement to a different physical location has the effect of creating an aircraft with a modified vector of attributes in the future. Decisions can also modify the attributes without moving the aircraft to another location (e.g. reconfiguring the aircraft).

We model the assignment of aircraft to demands as a network, with one node for each aircraft (more precisely, each type of aircraft) and one for each type of demand. From each aircraft node (these nodes may have a supply of 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.

The problem in Fig. 3 is a linear program which can be mathematically expressed as

$$x_t^n \;=\; \arg\max_{x_t \in \mathcal{X}_t^n} \left( \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} c_{tad} x_{tad} + \sum_{a' \in \mathcal{A}} \bar{V}_{ta'}^{n-1}(R_{ta'}^x) \right), \tag{11}$$

where $R_{ta}^x = R^{M,x}(R_{ta}^n, x_{ta})$ and the feasible set, $\mathcal{X}_t^n$, is defined by the flow conservation constraint,

$$\sum_{d \in \mathcal{D}} x_{tad} \;=\; R_{ta}^n, \tag{12}$$

as well as the requirement that a demand needs to be covered only once. $\bar{V}_{ta'}^{n-1}(R_{ta'}^x)$ is a piecewise linear function of the total flow into attribute node $a'$ at time $t$. The decision problem (11) is solved for a particular sample realization $\omega^n$. The solution to the problem is a vector, $x_t^n$. If the vector component $x_{tad} = 1$, then aircraft with

. **Step 0:** Initialize $\bar{V}_t^0(R_t)$ and pick an initial state $R_0$. 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:** Do for $t = 0, 1, \ldots, T-1$:
  . **Step 2.1:** Solve

$$x_t^n = \arg\max_{x_t \in \mathcal{X}_t^n} \left(C(R_t^n, x_t) + \bar{V}_t^{n-1}(R^{M,x}(R_t^n, x_t))\right) \tag{13}$$

  Let $\hat{v}_{ta}^n$ be the dual variable for constraint (12).
  . **Step 2.2** Update the value function using

$$\bar{V}_{t-1}^n(R_{t-1}^x) = U^V(\bar{V}_{t-1}^{n-1}(R_{t-1}^x), R_{t-1}^{x,n}, \hat{v}_t^n).$$

  . **Step 2.3:** Update the state of the system using

$$R_{t+1}^n = R^M(R_t^n, x_t, W_{t+1}(\omega^n)). \tag{14}$$

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

Fig. 4. The approximate dynamic programming algorithm

attribute $a$ uses decision $d$. The formulation is described in more detail in Appendix A.2.

### 3.3. The Basic ADP Algorithm

In Fig. 4, we sketch a simple ADP algorithm that incorporates the features discussed in the previous sections. At each iteration, the value function updates based on the information obtained. Proper design and careful updating of the value function approximation can produce optimizing behavior in the sense that the solution will tend to get better over the iterations. There is no guarantee, however, that the algorithm will produce the optimal solution, even in the limit.

We ran some initial calibration experiments to tune the stepsize rule and choose an appropriate number of iterations. We found that the solution stabilized nicely after 50 iterations, but we ran it for 100 iterations to be safe. Each forward pass over 120 time periods required 2-3 seconds per iterations, translating to run times of 3-5 minutes.

In step 2.1, we first solve a linear relaxation of the problem to obtain the dual variables. Our research has shown that for this class of approximations, the LP relaxation returns integer solutions 99.9% of the time (see [Topaloglu and Powell 2006]). If a fractional solution is found, we solve the problem again explicitly as an integer program.

## 4. EXPERIMENTS

In this section, we describe the experiments and results from a series of simulations that were run using data provided for a Canadian airlift operation. The goal of the research is to quantify the cost of uncertainty, and to determine how this cost is affected by the use of robust decisions as produced by our ADP algorithm.

The simulation considered the cost of flying aircraft, combined with penalties for serving demands outside of the specified time windows. Each movement request was assigned a priority from 1 (lowest) to 8, and a reward was assigned that increased with the priority of the movement. Coverage is an important metric, but we cannot guarantee that 100 percent of demands are moved on time. We use penalties for late movements and lost rewards for demands that are never moved to balance coverage and on-time performance against the cost of moving aircraft empty. In the experiments below we focus primarily on the objective function as a measure of solution quality that captures all these issues.

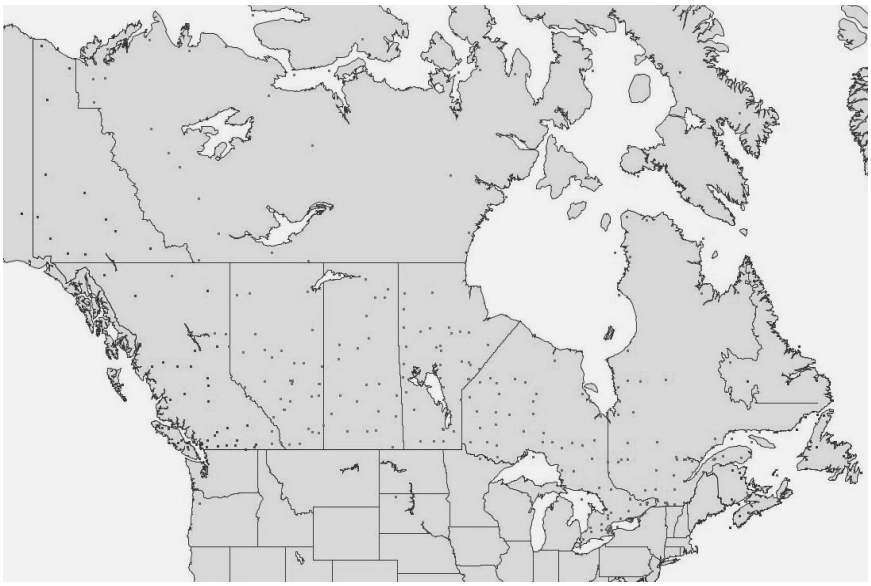

Fig. 5. Map of airfields in the Canadian airlift problem.

Our experiments progress in stages. We begin in section 4.1 by providing details regarding the design of the experiments. In section 4.2, we analyze deterministic datasets, where we focus on tuning the algorithm and developing an understanding of the behavior of the model. In section 4.3, we quantify the cost of uncertainty in customer demands using both myopic and robust decision-making behavior. Section 4.4 introduces the additional uncertainty of equipment failures.

### 4.1. Experimental setup

In this section, we describe the various parameters used in the experimental design of the airlift problem, using data provided by Defense Research (DRDC) of Canada. In our experiments, we considered a fleet of 13 aircraft, serving demands that could arise in 401 locations. Fig. 5 shows the locations of the airfields where the aircraft may move to serve demands. For the deterministic experiments, 171 demands were considered. For the stochastic case, we sampled the demands, choosing about 20 percent (in each iteration) from a pool of 3420 demands. The simulations were run for a period of $T = 720$ hours (one month), with decisions to assign aircraft to demands made every 6 hours (which means there are 120 time periods). A typical subproblem (see step 2.1 of Fig. 4) had about 10 aircraft and 70 demands. The objective function (2) includes the rewards for covering the demands, costs for moving the aircraft from one location to another and penalties for unserved demands. Each subproblem was solved to optimality using a commercial solver.

### 4.2. Deterministic demands

We first focused on a purely deterministic model to develop an understanding of both the original problem and the behavior of the ADP algorithm under different modeling parameters. Of particular importance is a parameter known as the planning horizon. This refers to the time-period over which we can anticipate resources will be available, enabling us to make decisions to act upon those resources. At time $t$, we allowed the model to assign aircraft to demands that were both actionable within a specified

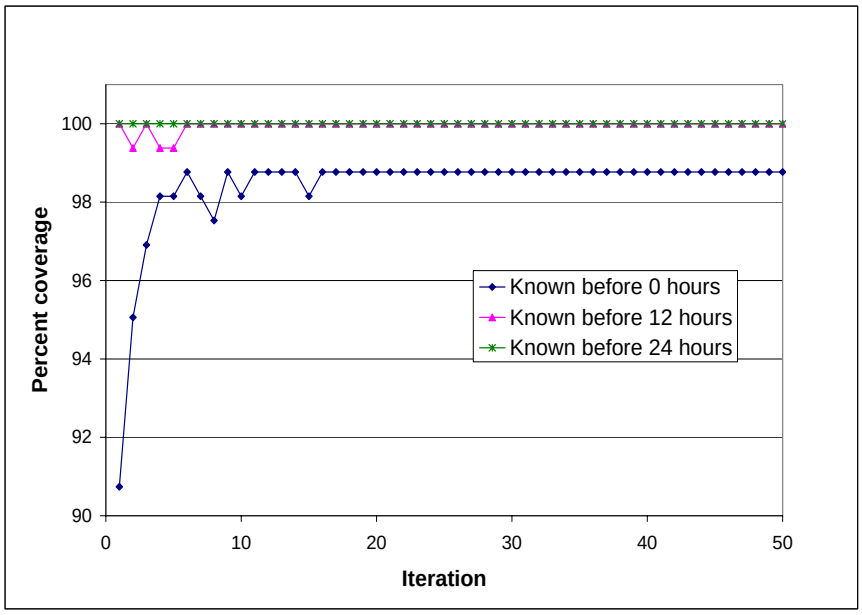


Fig. 6. Effect of planning horizon and ADP for a deterministic dataset, as a function of the number of iterations $n$ of the ADP algorithm.

planning horizon. A planning horizon of 0 means that we could only assign aircraft available now to requirements that could be moved right now. Fig. 6 shows the fraction of demands that were covered for planning horizons of 0, 12 and 24 hours, over the iterations of the ADP algorithm. The results show that the ADP algorithm produces a significant improvement in its ability to cover demands if the planning horizon is set to zero, achieving almost 99 percent coverage. When the planning horizon is lengthened to 12 or 24 hours, we obtain virtually 100 percent coverage even with a myopic policy. The results demonstrate that the value function approximations can produce desirable results for a deterministic dataset with no advance information.

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 demands 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 demands within the time windows which define when a demand is allowed to be served. Typically, in a stochastic, dynamic problem, we are trying to serve as many demands as we can, as quickly as possible. If we cannot serve all demands with available aircraft, we determine which ones to serve using a combination of assignment costs, the value of the demand (determined by its priority), and the downstream value function. It is the value function (which gives the value of the aircraft after the demand is completed) that helps us with the sequencing (for example, we can serve a short demand now and still cover another demand later). The execution times for the ADP algorithm 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 (see [Desrosiers et al. 1984], [Desrosiers et al. 1995], and [Powell and Chen 1999]). We can obtain very high quality solutions with tight time windows, or very wide time

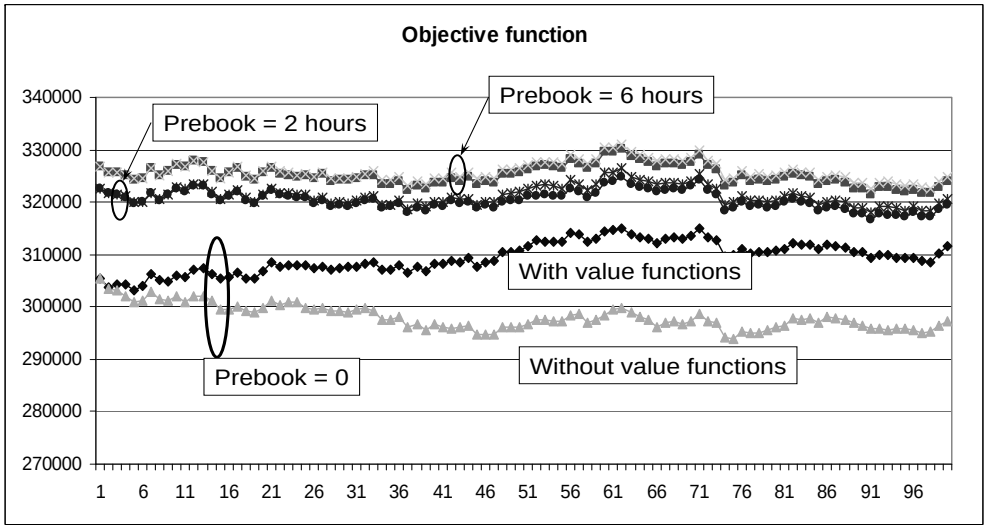


Fig. 7. Objective function for prebook times of 0,2 and 6 hours, with and without ADP.

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).

### 4.3. Random demands

The next set of experiments introduced uncertainty in the demands. Demands were divided between strategic demands (those known in advance) and dynamic demands which are revealed as the system evolves. We assumed that each dynamic demand was first learned at time $\tau^{pb}$ (the prebook time) before the beginning of the time window for the demand. 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 $t + \tau^{pb}$. We randomized the dynamic demands by taking the original set of dynamic demands, making five copies of each demand, and then randomly including each demand in a sample path with probability 0.2, so that the total number of demands remains roughly correct. In this way, we honestly do not know what demands will be in the system before the time at which each demand becomes known.

Fig. 7 shows the objective functions for prebook times of 0, 2 and 6 hours. Each run was performed with and without ADP. 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.

Fig. 8 gives the percentage improvement in the demand coverage (over a myopic policy) when the ADP procedure is used. 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).

An analysis of the actual demand coverage using each policy enabled us to determine the value of knowing demands in advance when demands were uncertain. The demand coverage obtained using the two policies is shown in Fig. 9. It is seen that the demand coverage improves as the prebook time is increased. The effect of the additional prebook time, however, is reduced significantly when we use approximate value functions. Without ADP, the improvement in the coverage as the prebook time increases from 0

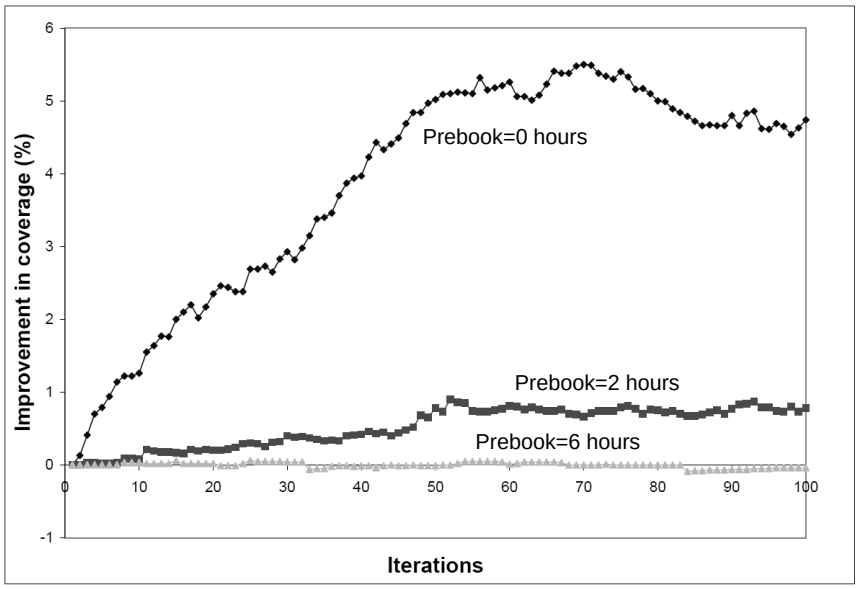


Fig. 8. Improvement in demand coverage as a result of ADP, for three different prebook times.

to 2 hours appears to rise from just over 90 percent to almost 98 percent. With ADP, the coverage increases from just under 96 percent to 98 percent.

These results demonstrate that the value of advance information is significant if we use a myopic policy, but is reduced dramatically when we use a robust policy obtained with ADP. Therefore, we feel that analyses that do not use robust decisions significantly overestimate the value of reducing uncertainty. These results make intuitive sense because if we assume a perfect future (say, we need three aircraft at a location), then we will plan accordingly, providing only what is precisely needed. A robust solution might provide additional aircraft, or might place aircraft at central locations where they can be dispatched quickly to regions where they are needed. Our results show that the ADP solution is less sensitive to uncertainty, so we feel that it is fair to conclude that ADP is producing a robust policy. It does not seem surprising, then, that if you use a robust policy, then reducing the level of uncertainty will have less of an impact than when you are assuming a perfect forecast.

### 4.4. Random aircraft failures

We next introduced the behavior that aircraft may fail at the end of each segment. The first set of results is shown in Fig. 10, which reports on the following runs: a) Aircraft which break down, and a myopic policy. b) Aircraft which break down, but using ADP. c) Aircraft do not break down, and a myopic policy (the objective function in this case does not change with the iterations - there is no random sampling of breakdowns, and no learning as there is with ADP). d) Aircraft do not break down, but using ADP. These runs assumed demands were deterministic, but dynamic demands became known with no advance warning. The results clearly show that the solution quality degrades in the presence of equipment failures, whether or not we use ADP.

Fig. 11 shows the objective function for all possible combinations: random and deterministic demands, with and without aircraft failures, and using a myopic policy or an ADP algorithm (eight runs in total). Throughout, dynamic demands remained with no

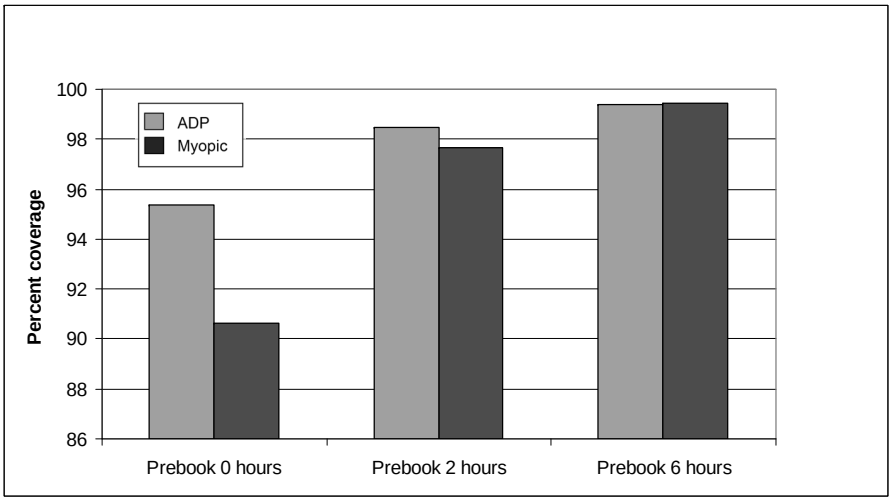


Fig. 9. The effect of robust decisions on the value of advance information.

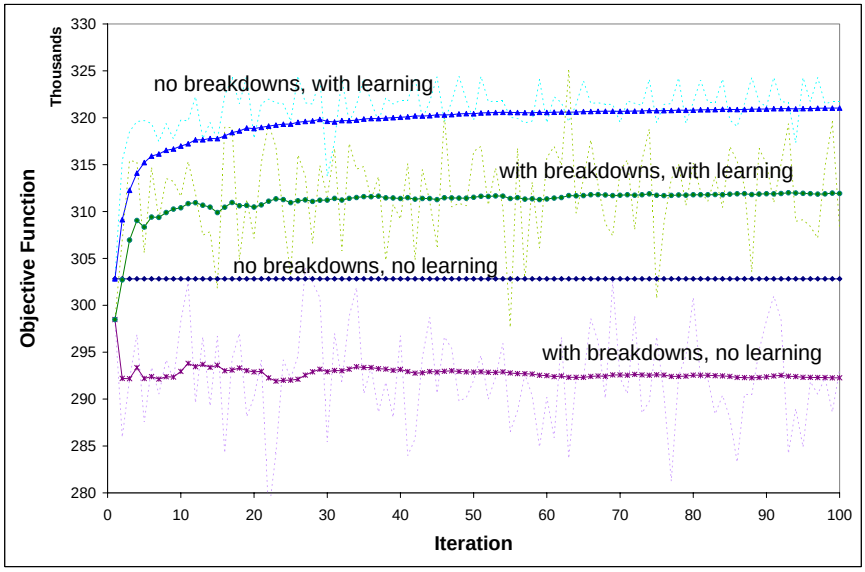


Fig. 10. Objective function in the presence/absence of breakdowns with and without learning using ADP.

advance warning. The runs are listed in the legend in the same order as the objective function at the last iteration.

These runs produce the following expected results. a) The best results (highest objective function) are obtained with deterministic demands, no breakdowns and with an ADP algorithm. b) The worst results are obtained with random demands, aircraft failures and no adaptive learning. c) Randomness in demands or aircraft failures produces worse results than without this source of randomness (and all else held equal). d) Learning using ADP always outperforms a myopic policy (with all else held equal).

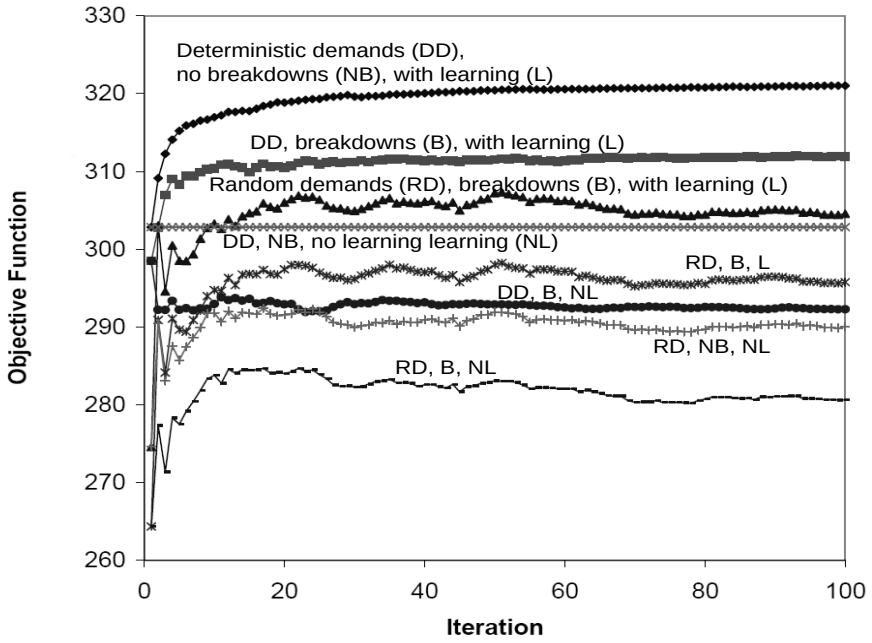


Fig. 11. Objective function for all combinations: DD - deterministic demands; RD - random demands; B - with aircraft failures; NB - without aircraft failures; L - using ADP; NL - myopic policy.

Other results are not obvious, and probably depend on the specific dataset. For our experiments, we found: e) Randomness in demands has a more negative impact than aircraft failures, with or without ADP. f) Randomness in demands with a policy using ADP outperforms deterministic demands with a myopic policy, when aircraft breakdowns are allowed (this result is especially surprising).

## 5. CONCLUSIONS

The experiments with the Canadian airlift problem accomplish two goals. First, we show that ADP produces consistently higher quality solutions in the presence of uncertainty than a myopic policy. We evaluated the policy in the presence of both random demands and equipment failures, obtaining consistent results in both cases. We note that our myopic policy is more sophisticated than the rule-based policies used in simulators by the airlift mobility command, and is a policy that is widely used in commercial software packages for solving real-time routing and scheduling problems.

Second, we show that the use of the type of robust policy produced by ADP lowers the cost of uncertainty, and therefore lowers the value of advance information. This result suggests that estimates of the value of knowing customer demands farther in advance may be significant less than those that would be estimated using an optimization model that does not properly handle the uncertainty of future events.

## ACKNOWLEDGMENTS

The research was supported by a grant from Defense Research (DRDC) of Canada. The simulations were performed using a software library developed under sponsorship the Air Force Office of Scientific Research, grant AFOSR-FA9550-05-1-0121.

## APPENDIX

### A.1. Evolution of the system

We use $\hat{R}_t$ to represent exogenous changes to the resource state vector that occur between $t-1$ and $t$. For example, it could include new customer demands or equipment failures (a plane that should have been available at time $t = 192$ will now be available at $t = 204$). We note that in a more general model, we can allow exogenous information (represented by $W_t$) that changes other elements of the problem than just the resource vector.

We model the evolution of the state of individual resources (the aircraft) using the ***attribute transition function*** $a_{t+1} = a^M(a_t, d_t, W_{t+1})$ which describes how the attribute vector changes as a result of new information, $W_{t+1}$, and a decision $d_t$ acting on a particular resource with attribute $a_t$. For example, suppose an aircraft at location J is acted upon by a decision to serve a demand in location K. This changes its location attribute from J to K. At the same time, its loaded/empty status is transformed from empty to loaded. For algebraic purposes, we also define an indicator function,

$$\delta_{a'}(a, d, W) = \begin{cases} 1 & \text{if } a^M(a, d, W) = a', \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

We can apply the attribute transition function to all the resources, which produces the ***resource transition function*** represented using $R_{t+1} = R^M(R_t, x_t, W_{t+1})$. We may write the evolution of the resource vector using

$$R_{t+1,a'} = \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} \delta_{a'}(a, d, W_{t+1}) x_{tad}. \tag{16}$$

The post-decision attribute state of a single resource captures the effect of the decision $d_t$, but not the effect of new information $W_{t+1}$. This would be written as $a_t^x = a^{M,x}(a_t, d_t)$. We may represent this transformation using an indicator function $\delta_{a'}(a, d)$ similar to the previous indicator function in equation (15).

$$\delta_{a'}(a, d) = \begin{cases} 1 & \text{if } a^{M,x}(a, d) = a', \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

This allows us to represent the evolution of the post-decision state as follows:

$$R^x_{ta'} = \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} \delta_{a'}(a, d) x_{tad}. \tag{18}$$

Using equations (16) and (18) and the notation for exogenous changes in the resource vector, we can establish the following relation between the pre- and post-decision resource vectors:

$$R_{t+1,a} = R^x_{ta} + \hat{R}_{t+1,a}. \tag{19}$$

### A.2. Formulation of the decision problem

We first define the following terms:

$$\begin{aligned} \mathcal{D}^D A &= \text{The set of decisions to serve demands.} \\ b_d &= \text{The demand type corresponding to decision } d \in \mathcal{D}^D. \end{aligned}$$

The feasible set for the decision problem has to satisfy the following constraints: There should be conservation of flows, and a demand can be covered only once. The complete

formulation of the decision problem is given in equation (11).

$$x_t^n = \arg\max_{x_t \in \mathcal{X}_t^n} \left( \sum_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} c_{tad} x_{tad} + \sum_{a' \in \mathcal{A}} \bar{V}_{ta'}^{n-1}(R_{ta'}^x) \right), \tag{20}$$

where $\mathcal{X}_t^n$ is given by

$$\sum_{d \in \mathcal{D}} x_{tad} = R_{ta}^{A,n}, \tag{21}$$

$$\sum_{a \in \mathcal{A}} x_{tad} \leq R_{t,b_d}^{D,n}, \quad d \in \mathcal{D}^D, \tag{22}$$

$$x_{tad} \geq 0 \quad \text{and integer}. \tag{23}$$

The elements of the post-decision state resource vector $R_{ta'}^x$ are computed using equation (18). Equations (21) and (22) represent the flow conservation constraint and the constraint that a demand can be covered only once, respectively. The latter constraint applies only to those decisions that represent serving demands, that is, each $d \in \mathcal{D}^D$ for which there is a corresponding demand type, $b_d \in \mathcal{B}$.

## ACKNOWLEDGMENTS

The authors would like to thank the careful comments of the reviewers which helped improve the presentation.

## REFERENCES

BAKER, S., MORTON, D., ROSENTHAL, R., AND WILLIAMS, L. 2002. Optimizing Military Airlift. *Operations Research 50*, 582–602.

BERTSEKAS, D. P. 2009. *Approximate Dynamic Programming* 3 Ed. Vol. II. Athena Scientific, Belmont, MA, Chapter 6.

BERTSEKAS, D. P. AND TSITSIKLIS, J. N. 1996. *Neuro-dynamic programming*. Athena Scientific, Belmont, MA.

BURKE, J. F. J., LOVE, R. J., AND MACAL, C. M. 2004. Modelling Force Deployments from Army Installations Using the Transportation System Capability (TRANSCAP) Model: A Standardized Approach. *Mathematical and Computer Modelling 39*, 733–744.

CRINO, J. R., MOORE, J. T., BARNES, J. W., AND NANRY, W. P. 2004. Solving the Theater Distribution Vehicle Routing and Scheduling Problem Using Group Theoretic Tabu Search. *Mathematical and Computer Modelling 39*, 599–616.

DANTZIG, G. AND FERGUSON, A. 1956. The Allocation of Aircrafts to Routes: An Example of Linear Programming Under Uncertain Demand. *Management Science 3*, 45–73.

DESROSIERS, J., DUMAS, Y., SOLOMON, M. M., AND SOUMIS, F. 1995. *Time Constrained Routing and Scheduling*. North Holland, Amsterdam, 35–139.

DESROSIERS, J., SOUMIS, F., AND DESROCHERS, M. 1984. Routing with time windows by column generation. *Networks 14,* 4, 545–565.

GEORGE, A. P. AND POWELL, W. B. 2006. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Journal of Machine Learning 65,* 1, 167–198.

GOGGINS, D. A. 1995. Stochastic Modeling for Airlift Mobility. M.S. thesis, Monterey, CA.

KILLINGSWORTH, P. S. AND MELODY, L. J. 1997. Should C17's be Deployed as Theater Assets?: An Application of the CONOP Air Mobility Model.

LAGOUDAKIS, M. AND PARR, R. 2003. Least-squares policy iteration. *Journal of Machine Learning Research 4*, 1149.

LIST, G. F., WOOD, B., NOZICK, L. K., TURNQUIST, M. A., JONES, D. A., KJELDGAARD, E. A., AND LAWTON, C. R. 2003. Robust optimization for fleet planning under uncertainty. *Transportation Research 39*, 209–227.

MENACHE, I., MANNOR, S., AND SHIMKIN, N. 2005. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research 134,* 1, 215–238.

MIDLER, J. L. AND WOLLMER, R. D. 1969. Stochastic Programming Models for Airlift Operations. *Naval Research Logistics Quarterly 16*, 315–330.

MORTON, D., SALMERON, J., AND WOOD, R. 2003. A stochastic program for optimizing military sealift subject to attack. *Stochastic Programming E-Print Series*.

MULVEY, J. M., VANDERBEI, R. J., AND ZENIOS, S. A. 1995. Robust Optimization of Large-Scale Systems. *Operations Research 43,* 2, 264–281.

NIEMI, A. 2000. *Stochastic modeling for the NPS/RAND Mobility Optimization Model*. Available: http://ie.engr.wisc.edu/robinson/Niemi.htm.

POWELL, W. B. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, Hoboken, NJ.

POWELL, W. B. AND CHEN, Z.-L. 1999. Solving Parallel Machine Scheduling Problems by Column Generation. *Informs Journal on Computing 11*, 78–94.

POWELL, W. B. AND GODFREY, G. 2002. An adaptive dynamic programming algorithm for dynamic fleet management, I: Single period travel times. *Transportation Science 36,* 1, 21–39.

POWELL, W. B. AND GODFREY, G. A. 2001. An adaptive, distribution-free approximation for the newsvendor problem with censored demands, with applications to inventory and distribution problems. *Management Science 47,* 8, 1101–1112.

POWELL, W. B., RUSZCZYNSKI, A., AND TOPALOGLU, H. 2004. Learning Algorithms for Separable Approximations of Discrete Stochastic Optimization Problems. *Mathematics of Operations Research 29,* 4, 814–836.

POWELL, W. B. AND TOPALOGLU, H. 2005. Fleet Management. In *Applications of Stochastic Programming*, S. Wallace and W. Ziemba, Eds. Math Programming Society - SIAM Series in Optimization, Philadelphia, 185–216.

POWELL, W. B. AND VAN ROY, B. 2004. Approximate Dynamic Programming for High Dimensional Resource Allocation Problems. In *Handbook of Learning and Approximate Dynamic Programming*, J. Si, A. G. Barto, W. B. Powell, and D. W. II, Eds. IEEE Press, New York.

PUTERMAN, M. L. 1994. *Markov Decision Processes* 1st Ed. Johnn Wiley and Sons, Hoboken.

ROSENTHAL, R., MORTON, D., BAKER, S., LIM, T., FULLER, D., GOGGINS, D., TOY, A., TURKER, Y., HORTON, D., AND BRIAND, D. 1997. Application and Extension of The Thruput II Optimization Model For Airlift Mobility. *Military Operations Research 3*, 55–74.

SUTTON, R. S. AND BARTO, A. G. 1998. *Reinforcement Learning*. Vol. 35. MIT Press, Cambridge, MA.

SZEPESVARI, C. 2010. *Algorithms for Reinforcement Learning*. Vol. 4. Morgan and Claypool.

TOPALOGLU, H. AND POWELL, W. B. 2003. An Algorithm for Approximating Piecewise Linear Concave Functions from Sample Gradients. *Operations Research Letters 31*, 66–76.

TOPALOGLU, H. AND POWELL, W. B. 2006. Dynamic Programming Approximations for Stochastic, Time-Staged Integer Multicommodity Flow Problems. *Informs Journal on Computing 18*, 31–42.

WING, V. F., RICE, R. E., SHERWOOD, R., AND ROSENTHAL, R. E. 1991. Determining the Optimal Mobility Mix. Tech. rep., Force Design Division, The Pentagon, Washington D.C.

WU, T., POWELL, W. B., AND WHISMAN, A. 2009. The Optimizing-Simulator: An Illustration using the Military Airlift Problem. *ACM Transactions on Modeling and Simulation 19,* 3, 1–31.

YOST, K. A. 1994. The THRUPUT Strategic Airlift Flow Optimization Model. Tech. rep., Air Force Studies and Analyses Agency, The Pentagon, Washington D.C.