The Optimizing-Simulator: An Illustration Using the Military Airlift Problem

TONGQIANG TONY WU and WARREN B. POWELL
Princeton University
and
ALAN WHISMAN
Air Mobility Command Retired

There have been two primary modeling and algorithmic strategies for modeling operational problems in transportation and logistics: simulation, offering tremendous modeling flexibility, and optimization, which offers the intelligence of math programming. Each offers significant theoretical and practical advantages. In this article, we show that you can model complex problems using a range of decision functions, including both rule-based and cost-based logic, and spanning different classes of information. We show how different types of decision functions can be designed using up to four classes of information. The choice of which information classes to use is a modeling choice, and requires making specific choices in the representation of the problem. We illustrate these ideas in the context of modeling military airlift, where simulation and optimization have been viewed as competing methodologies. Our goal is to show that these are simply different flavors of a series of integrated modeling strategies.

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

Additional Key Words and Phrases: Approximate dynamic programming, military logistics, control of simulation, modeling information, optimizing-simulator

ACM Reference Format:
Wu, T. T., Powell, W. B., and Whisman, A. 2009. The optimizing-simulator: An illustration using the military airlift problem. ACM Trans. Model. Comput. Simul., 19, 3, Article 14 (June 2009), 31 pages. DOI = 10.1145/1540530.1540535 http://doi.acm.org/ 10.1145/1540530.1540535

This research was supported in part by grant AFOSR contract FA9550-08-1-0195.
Authors' addresses: T. T. Wu and W. B. Powell, Dept. of Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544; email: powell@princeton.edu; A. Whisman, Air Mobility Command, Scott Air Force Base, IL 62225.

DOI 10.1145/1540530.1540535 http://doi.acm.org/10.1145/1540530.1540535

1. INTRODUCTION

Complex operational problems, such as those that arise in transportation and logistics, have long been modeled using simulation or optimization. Typically, these are viewed as competing approaches, each offering benefits over the other. Simulation offers significant flexibility in the modeling of complex operational conditions, and in particular is able to handle various forms of uncertainty. Optimization offers intelligent decisions that often allow models to adapt quickly to new datasets (simulation models often require recalibrating decision rules), and offer additional benefits such as dual variables. In the desire for good solutions, optimization seeks to find the best solution, but typically requires making a number of simplifications.

We encountered the competition between simulation and optimization in the context of modeling the military airlift problem faced by the analysis group at the Airlift Mobility Command (AMC). The military airlift problem deals with effectively routing a fleet of aircraft to deliver loads of people and goods (troops, equipment, food, and other forms of freight) from different origins to different destinations as quickly as possible under a variety of constraints. In the parlance of military operations, loads (or demands) are referred to as requirements. Cargo aircraft come in a variety of sizes, and it is not unusual for a single requirement to need multiple aircraft. If the requirement includes people, the aircraft has to be configured with passenger seats. Other issues include maintenance, airbase capacity, weather, and the challenge of routing aircraft through friendly airspaces.

There are two major classes of models that have been used to solve the military airlift (and closely related sealift) problem: cost-based optimization models [Morton et al. 1996; Rosenthal et al. 1997; Baker et al. 2002], and rule-based simulation models, such as MASS (Mobility Analysis Support System) and AMOS (Air Mobility Operations Simulator), which are heavily used within the AMC. The analysis group at AMC has preferred AMOS because it offers tremendous flexibility, as well as the ability to handle uncertainty. However, simulation models require that the user specify a series of rules to obtain realistic behaviors. Optimization models, on the other hand, avoid the need to specify various decision rules, but they force the analyst to manipulate the behavior of the model (the decisions that are made) by changing the objective function. While optimal solutions are viewed as the gold standard in modeling, it is a simple fact that for many applications, objective functions are little more than coarse approximations of the goals of an operation.

Our strategy should not be confused with simulation-optimization, which is well-known within the simulation community (see, for example, the excellent reviews Swisher et al. [2003] and Fu [2002]). This strategy typically assumes an often myopic parameterized policy, where the goal is to find the best settings for one or more parameters. For our class of applications, we are trying to construct a simulation model that can directly compete with a math programming model, while retaining many of the important features that classical simulation methods bring to the table. For example, we need decisions that consider their impact on the future. At the same time, we need a model that will handle uncertainty and a high level of detail, features that we take for granted in simulation models.

This article proposes to bring together the simulation and optimization communities who work on transportation and logistics. We do this by combining math programming, approximate dynamic programming, and simulation, with a strong dependence on machine learning. From the perspective of the simulation community, it will look as if we are running a simulation iteratively, during which we can estimate the value of being in a state (dynamic programming), and we can also measure the degree to which we are matching exogenously specified patterns of behavior (a form of supervisory control from the reinforcement-learning community). At each point in time, we use math programming to solve sequences of deterministic optimization problems. Math programming allows us to optimize at a point in time, while dynamic programming allows us to optimize over time. The pattern matching allows us to bridge the gap between cost-based optimization and rule-based simulation.

This strategy has the effect of allowing us to build a family of decision functions with up to four classes of information: (1) the physical state (what we know about the system now), (2) forecasts of exogenous information events (new customer demands, equipment failures, weather delays), (3) forecasts of the impact of decisions now on the future (giving rise to value functions used in dynamic programming), and (4) forecasts of decisions (which we represent as patterns). The last three classes of information are all a form of forecast. If we just use the first class, we get a classical simulation using a myopic policy, although these come in two flavors: rule-based (popular in simulation) and cost-based (popular in the transportation community when solving dynamic problems). If we use the second information class, we obtain a rolling-horizon procedure. The third class of information uses approximate dynamic programming to estimate the value of being in a state. The fourth class allows us to combine cost-based logic (required for any optimization model) with particular types of rules (for example, “we prefer to use C-17s for loads originating in Europe”). This class introduces the use of proximal point algorithms. We claim that any existing modeling and algorithmic strategy can be classified in terms of its use of these four classes of information.

The central contribution of this article is to identify how simulation and optimization can be combined to address complex modeling problems that arise in transportation and logistics, illustrated using the context of a military airlift application. This problem class has traditionally been solved using classical deterministic optimization methods or simulation, with each approach having significant strengths and weaknesses. We show how cost-based and rule-based logic can be combined within this broad framework. We illustrate these ideas using an actual airlift simulation to show that we can vary the information content of decisions to produce decisions with increasing levels of sophistication.

We begin in Section 2 with a review of the field of modeling military mobility problems (this is how this community refers to this problem class). More than just a literature review, this section allows us to contrast the modeling styles of different communities, including deterministic math programming, simulation, and stochastic programming. In Section 3 we provide our own model of the airlift problem, providing only enough notation to illustrate the important modeling principles. Section 4 shows how we can create different decision functions by modeling the information available to make a decision. We also discuss rule-based and cost-based functions, and show how these can be integrated into a single, general decision function that uses all four classes of information. In Section 5 we simulate all the different information classes, and show that as we increase the information (that is, use additional information classes) we obtain better solutions, measured in terms of throughput, a common measure used in the military, and realism—reflected by our ability to match desired patterns of behavior. Section 6 concludes the article.

2. MODELING OF MILITARY MOBILITY PROBLEMS

This section provides a summary of different modeling strategies for military mobility problems: air and sea. After providing a review of the military mobility literature in Section 2.1, we briefly summarize the three primary modeling strategies that have been used in this area: deterministic linear programming (Section 2.2), simulation (Section 2.3), and stochastic programming (Section 2.4). The modeling of mobility problems is unusual in that it has been approached in detail by all three communities. We present these models in only enough detail to allow us to contrast the different modeling styles.

2.1 The History of Mobility Modeling

Ferguson and Dantzig [1955] is one of the first to apply mathematical models to air-based transportation. Subsequently, numerous studies were conducted on the application of optimization modeling to the military airlift problem. Several mathematical modeling formulations for military airlift operations are described by Mattock et al. [1995]. The RAND Corporation published a very extensive analysis of airfield capacity in Stucker and Berg [1999]. According to Baker et al. [2002], research on air mobility optimization at the Naval Postgraduate School (NPS) started with the Mobility Optimization Model (MOM). This model is described in Wing et al. [1991] and concentrates on both sealift and airlift operations. Therefore, the MOM model is not designed to capture the characteristics specific to airlift operations, but it is a good model in the sense that it is time-dependent. THRUPUT, a general airlift model developed by Yost [1994], captures the specifics of airlift operations but is static. The United States Air Force Studies and Analyses Agency in the Pentagon asked NPS to combine the MOM and THRUPUT models into one model that would be time dependent and would also capture the specifics of airlift operations [Baker et al. 2002]. The resulting model is called THRUPUT II, described in detail in Rosenthal et al. [1997]. During the development of THRUPUT II at NPS, a group at RAND developed a similar model called CONOP (CONcept of OPerations), described in Killingsworth and Melody [1997]. The THRUPUT II and CONOP models each possessed several features that the other lacked. Therefore, the Naval Postgraduate School and the RAND Corporation merged the two models into NRMO (NPS/RAND Mobility Optimizer), described in Baker et al. [2002].

Crino et al. [2004] introduced the group-theoretic tabu search method to solve the theater distribution vehicle routing and scheduling problem. Their heuristic methodology evaluates and determines the routing and scheduling of multi-modal theater transportation assets at the individual asset operational level.

A number of simulation models have also been proposed for airlift problems (and related problems in military logistics). Schank et al. [1991] review several deterministic simulation models. Burke et al. [2004] at the Argonne National Laboratory developed a model called TRANSCAP (Transportation System Capability) to simulate the deployment of forces from Army bases. The heart of TRANSCAP is the discrete-event simulation module developed in MODSIM III. Perhaps the most widely used model at the Air Mobility Command is AMOS which is a discrete-event worldwide airlift simulation model used in strategic and theater operations to deploy military and commercial airlift assets. It was once the standard for all airlift problems in AMC, and all airlift studies were compared with the results produced by the AMOS model. AMOS provides for a very high level of detail, allowing AMC to run analyses on a wide variety of issues.

One feature of simulation models is their ability to handle uncertainty, and as a result there has been a steady level of academic attention toward incorporating uncertainty into optimization models. Dantzig and Ferguson [1956] is one of the first to study uncertain customer demands in the airlift problem. Midler and Wollmer [1969] also takes into account stochastic cargo requirements. This work formulates two two-stage stochastic linear programming models: a monthly planning model, and a daily operational model for the flight scheduling. Goggins [1995] extended Throughput II [Rosenthal et al. 1997] to a two-stage stochastic linear program to address the uncertainty of aircraft reliability. Niemi [2000] expands the NRMO model to incorporate stochastic ground times through a two-stage stochastic programming model. To reduce the number of scenarios for a tractable solution, the model assumes that the set of scenarios is identical for each airfield and time period and a scenario is determined by the outcomes of repair times of different types of aircraft. The resulting stochastic programming model has an equivalent deterministic linear programming formulation. Granger et al. [2001] compared the simulation model and the network approximation model for the impact of stochastic flying times and ground times on a simplified airlift network (one origin aerial port of embarkation (APOE), three intermediate airfields and one destination aerial port of debarkation (APOD)). Based on the study, they suspect that a combination of simulation and network optimization models should yield much better performance than either one of these alone. Such an approach would use a network model to explore the variable space and identify parameter values that promise improvements in system performance, then validate these by simulation. Morton et al. [2002] developed the Stochastic Sealift Deployment Model (SSDM), a multi-stage stochastic programming model to plan the wartime, sealift deployment of military cargo subject to stochastic attack. They used scenarios to represent the possibility of random attacks.

A related method in Yost and Washburn [2000] combines linear programming with partially observable Markov decision processes to a military attack problem in which aircraft attack targets in a series of stages. They assume that the expected value of the reward (destroyed targets) and resource (weapon) consumption are known for each policy where a policy is chosen in a finite feasible set. Their method requires that the number of possible states of each object be small and the resource constraints be satisfied on the average. In the military airlift problem, the number of possible states is very large, as are the number of actions and outcomes. As a result, we could not have the expected value of the reward before we solve the problem. In general, we require that the resource constraints are strictly satisfied in a military airlift problem.

2.2 The NRMO Model

The NRMO model has been in development since 1996 and has been employed in several airlift studies. For a detailed review of the NRMO model, including a mathematical description, see Baker et al. [2002]. The goal of the NRMO model is to move equipment and personnel in a timely fashion from a set of origin bases to a set of destination bases using a fixed fleet of aircraft with differing characteristics. This deployment is driven by the movement and delivery requirements specified in a list called the Time-Phased Force Deployment Document (or Dataset) (TPFDD). This list essentially contains the cargo and troops, along with their attributes, that must be delivered to each of the bases of a given military scenario.

Aircraft types for the NRMO runs reported by Baker et al. [2002] include C-5, C-17, C-141, Boeing 747, KC-10, and KC-135. Different types of aircraft have different features, such as passenger and cargo-carrying capacity, airfield capacity consumed, range-payload curve, and so on. The range-payload curve specifies the weight that an aircraft can carry given a distance traveled. These range-payload curves are piecewise linear concave.

The activities in NRMO are represented using three time-space networks: the first one flows aircraft, the second the cargo (freight and passengers) and the third flows crews. Cargo and troops are carried from the onload aerial port of embarkation (APOE) to either the offload aerial port of debarkation (APOD) or the forward operating base (FOB). Certain requirements need to be delivered to the FOB via some other means of transportation after being offloaded at the APOD. Some aircraft, however, can bypass the APOD and deliver directly to the FOB. Each requirement starts at a specific APOE dictated by the TPFDD and then is either dropped off at an APOD or a FOB.

NRMO makes decisions about which aircraft to assign to which requirement, which freight will move on an aircraft, which crew will handle a move, as well as variables that manage the allocation of aircraft between roles such as long range, intertheater operations and shorter, shuttle missions within a theater. The model can even handle the assignment of KC-10s between the role of strategic cargo hauler and midair refueler.

In the model, NRMO minimizes a rather complex objective function that is based on several costs assigned to the decisions. There are a variety of costs, including hard operating costs (fuel, maintenance) and soft costs to encourage desirable behaviors. For example, NRMO assigns a cost to penalize deliveries of cargo or troops that arrive after the required delivery date. This penalty structure charges a heavier penalty the later the requirement is delivered. Another term penalizes the cargo that is simply not delivered. The objective also penalizes reassignments of cargo and deadheading crews. Last, the objective function offers a small reward for planes that remain at an APOE, as these bases are often in the continental US and are therefore close to the home bases of most planes. The idea behind this reward is to account for uncertainty in the world of war and to keep unused planes well positioned in case of unforeseen contingencies.

The constraints of the model can be grouped into seven categories: (1) demand satisfaction; (2) flow balance of aircraft, cargo and crews; (3) aircraft delivery capacity for cargo and passengers; (4) the number of shuttle and tanker missions per period; (5) initial allocation of aircraft and crews; (6) the usage of aircraft of each type; and (7) aircraft handling capacity at airfields. A general statement of the model (see Baker et al. [2002] for a detailed description) is given by:

$$\min_{(x_t),t=0,\ldots,T} \sum_{t=0}^{T} c_t x_t \tag{1}$$

subject to

$$A_t x_t - \sum_{\tau=1}^{t} B_{t-\tau,t} x_{t-\tau} \;=\; R_t, \quad t = 0, 1, \ldots, T, \tag{2}$$

$$D_t x_t \;\le\; u_t, \quad t = 0, 1, \ldots, T, \tag{3}$$

$$x_t \;\ge\; 0, \quad t = 0, 1, \ldots, T, \tag{4}$$

where,

t = the time at which an activity begins,
τ = the time required to complete an action,
A_t = incidence matrix giving the elements of x_t that represent departures from time t,
$B_{t-\tau,t}$ = incidence matrix giving the elements of x_t that represent flows arriving at time t,
D_t = incidence matrix capturing flows that are jointly constrained,
u_t = upper bounds on flows,
R_t = the resources available at time t,
x_t = the decision vector at time t, where an element might be x_{tij} telling us if aircraft i is assigned to requirement j at time t,
c_t = the cost vector at time t.

NRMO is implemented with the algebraic modeling language GAMS [Brooke et al. 1992], which facilitates the handling of states. Each state is identified by an index combination of elements drawn from sets of aircraft attributes such as time periods, requirements, cargo types, aircraft types, bases, and routes.

An example of an index combination would be an aircraft of a certain type delivering a certain requirement on a certain route departing at a certain time. Only feasible index combinations are considered in the model so that the computation becomes tractable. The information in the NRMO model is captured in the TPFDD, which is known at the beginning of the horizon.

NRMO requires that the behavior of the model be governed by a cost model (which simulators do not require). The use of a cost model minimizes the need for extensive tables of rules to produce good behaviors. The optimization model also responds in a natural way to changes in the input data (for example, an increase in the capacity of an airbase will not produce a decrease in overall throughput). But linear programming formulations suffer from weaknesses. A significant limitation is the difficulty in modeling complex system dynamics. For example, a simulation model can include logic such as, "if there are four aircraft occupying all the parking spaces, a fifth aircraft will have to be pulled off the runway where it cannot be refueled or repaired." In addition, linear programming models cannot directly model the evolution of information. This limits their use in analyzing strategies that directly affect uncertainty (What is the impact of last-minute requests on operational efficiency? What is the cost of sending an aircraft through an airbase with limited maintenance facilities, where the aircraft might break down?).

2.3 The AMOS Model

AMOS is a rule-based simulation model. A rough approximation of the rules proceeds as follows. AMOS starts with the first available aircraft, and then tries to see if the aircraft can move the first requirement that needs to be moved (there is logic to check if there is an aircraft at the origin of the requirement, but otherwise the distance the aircraft has to travel is ignored). Given an aircraft and requirement, the next problem is to evaluate the assignment. For simple transportation models, this step is trivial (e.g. a cost per mile times the number of miles). For more complex transportation problems (managing drivers), it is necessary to go through a more complex set of calculations that depend on the hours of service. For the military airlift problem, moving an aircraft from, say, the East Coast to India, requires moving through a sequence of intermediate airbases that have to be checked for capacity availability. This step is fairly expensive, so it is difficult to evaluate all possible combinations of aircraft and requirements. If AMOS determines that an assignment is infeasible, it simply moves to the next aircraft in the list.

We note that it is traditional to describe simulation models such as this with virtually no notation. We can provide a very high level notational system by defining:

S_t = the state vector of the system at time t, giving what we know about the system at time t, such as the current status of aircraft and requirements;

$X^\pi(S_t)$ = the function that returns a decision, x_t, given information S_t, when we are using policy π, where π is simply an index identifying the specific function being used;

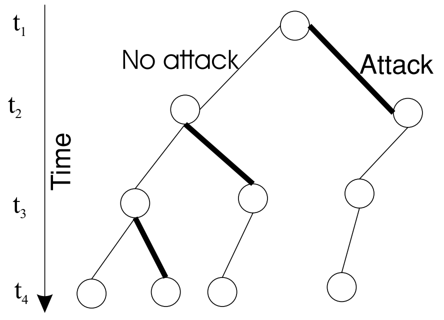


Fig. 1. The single attack scenario tree used in SSDM.

x_t	=	the vector of decisions at time t (just as in the NRMO model);
W_t	=	exogenous information arriving between $t-1$ and t;
$S^M(S_t, x_t, W_{t+1})$	=	the transition function (sometimes called the system model) that gives the state S_{t+1}, at time $t+1$, given that we are in state S_t, make decision x_t, and observe new exogenous information W_{t+1}.

Given a policy (that is, a decision function $X^\pi(S_t)$), a simulation model can be viewed as consisting of nothing more than the two equations:

$$x_t \leftarrow X^\pi(S_t), \tag{5}$$

$$S_{t+1} \leftarrow S^M(S_t, x_t, W_{t+1}(\omega)). \tag{6}$$

A more complete model would specify the state variable and transition function in greater detail. Note that we do not have an objective function.

2.4 The SSDM Model

The Stochastic Sealift Deployment Model (SSDM) [Morton et al. 2002] is a multistage stochastic mixed-integer program designed to hedge against potential enemy attacks on seaports of debarkation (SPODs) with probabilistic knowledge of the time, location and severity of the attacks. They test the model on a network with two SPODs (the number actually used in the Gulf War) and other locations such as a seaport of embarkation (SPOE) and locations near SPODs where ships wait to enter berths. To keep the model computationally tractable, they assumed only a single biological attack can occur. Thus, associated with each possible outcome (scenario in the language of stochastic programming) is whether or not an attack occurs. We assume that there is a finite set of scenarios Ω, and we let $\omega \in \Omega$ be a single scenario. If scenario ω occurs, $t(\omega)$ is the time at which it occurs. They then let $\mathcal{T}(\omega) = \{0, \ldots, t(\omega) - 1\}$ be the set of time periods preceding an attack that occurs at time $t(\omega)$. This structure results in a scenario tree that grows linearly with the number of time periods, as illustrated in Figure 1. The stochastic programming model for SSDM can then be written:

$$\min_{x_t(\omega), t=0,\ldots,T, \omega\in\Omega} \sum_{\omega\in\Omega} \sum_{t=0}^{T} p(\omega) c_t(\omega) x_t(\omega), \tag{7}$$

subject to

$$\begin{aligned}
A_t x_t(\omega) - \sum_{\tau=1}^{t} \Delta_{t-\tau,t} x_{t-\tau}(\omega) &= \hat{R}_t(\omega) \;\; t = 0, 1, \ldots, T, \;\; \omega \in \Omega, \\
B_t x_t(\omega) &\le u_t(\omega), \;\; t = 0, 1, \ldots, T, \;\; \omega \in \Omega, \\
x_t(\omega) &= x_t(\omega'), \;\; \text{for all } t \in \mathcal{T}(\omega) \cap \mathcal{T}(\omega'), \;\; \omega \in \Omega, \qquad (8) \\
x_t(\omega) &\ge 0 \;\text{ and integer, for all } t = 0, 1, \ldots, T, \;\; \omega \in \Omega.
\end{aligned}$$

Here ω is the index for scenarios. The probability of scenario ω, is given by $p(\omega)$. The variables t, τ, A_t, $\Delta_{t-\tau,t}$, B_t, and $\hat{R}_t$ are the same as introduced in the NRMO model. We let $u_t(\omega)$ be the upper bound on flows, where the capacities of SPOD cargo handling are varied under different attack scenarios. The cost vector for scenario ω is given by $c_t(\omega)$, and $x_t(\omega)$ is the decision vector. We let $\hat{R}_t(\omega)$ be the exogenous supply of aircraft that first become known at time t. The nonanticipativity constraints (8) say that if scenarios ω and ω' share the same branch on the scenario tree at time t, the decision at time t should be the same for those scenarios.

In general, the stochastic programming model is much larger than a deterministic linear programming model and still struggles with the same difficulty in modeling complex system dynamics. Multistage stochastic programming notoriously explodes in size when there are multiple stages, and multiple scenarios per stage. This particular problem exhibited enough structure to make it possible to construct a model that was computationally tractable. As a result, the SSDM model was able to make it possible to formulate the optimization problem while capturing the uncertainty in the potential attacks.

2.5 An Overview of the Optimizing-Simulator

In the remainder of this article, we are going to describe a way to optimize complex problems using the same framework as that described by Equations (5) and (6). The only difference will be in the construction of the decision function $X^\pi(S_t)$. In many simulation models, this function consists of a series of rules. Typically, these rules are designed to mimic how decisions are actually made, and there is no attempt to find the best decisions.

It is easy to envision a decision function that is actually a linear program, where we optimize the use of resources using what we know at a point in time, but ignoring the impact of current decisions on the future. This is an example of a cost-based policy, and like a rule-based policy, it would also be called a myopic policy, because it ignores the future. Alternatively, we could optimize over a horizon $t, t+1, \ldots, t+T$, and then implement the decision we choose at time t. This is classically known as a rolling-horizon procedure. There are other techniques to build ever more sophisticated decision functions that produce both more optimal behaviors (as measured by an objective function) and more realistic behaviors (as measured by the judgment of an expert user).

As indicated by Equations (5) and (6), the decision function (which contains the optimizing logic) and the transition function (which contains the simulation logic) communicate primarily through the state variable. The decision function uses the information in the state variable to make a decision. The transition function uses the state variable, the decision, and a sample of new information, to compute the state variable at the next point at which we will make a decision. We note that t indexes the times at which we make decisions. Physical events (the movement of people, equipment, and other resources) and the arrival of information (phone calls, equipment failures, price changes) all occur in continuous time.

We are going to illustrate the optimizing-simulator in the context of a military logistics problem defined over a finite planning horizon. The most sophisticated logic requires that this process be run iteratively, where the decision functions adaptively learn better behaviors. This logic is implemented using the framework of approximate dynamic programming, where at each iteration we are updating our estimates of the value of being in a state (for example, the value of having a C-5 in Europe). The logic scales for very large-scale problems, because the optimization logic never tries to simultaneously optimize over all time periods (as is done in linear programming models such as NRMO). Instead, the logic solves sequences of much smaller optimization problems, which makes it possible to scale to very large applications.

3. MODELING THE MILITARY AIRLIFT PROBLEM

We adopt the convention that decisions are made in discrete time, while information and physical events occur in continuous time. Time $t = 0$ represents the current time. Any variable indexed by t has access to all the information that has arrived prior to time t. By assuming that information arrives in continuous time, we remove any ambiguity about the measurability of any random variables. We model physical processes in continuous time since this is the most appropriate model (we derive no benefits from modeling physical events in discrete time). For transportation problems, decision epochs (the points in time at which decisions are made) are always modeled using uniform time intervals (e.g. every 4 hours) since transportation problems are always nonstationary, and there are numerous parallel events happening over networks. We are going to assume that we are modeling the problem over a finite horizon defined by the decision epochs $\mathcal{T} = \{0, 1, \ldots, T\}$.

We divide our model in terms of the resources involved in the problem (aircraft and requirements), exogenous information, decisions, the transition function, and functions used to evaluate the solution.

Aircraft and requirements.

$$\begin{aligned} a &= \text{The attribute vector describing an aircraft, such as} \\ &= \begin{bmatrix} \text{C-17(aircraft type)} \\ \text{50 Tons(capacity)} \\ \text{EDAR(current airbase)} \\ \text{40 Tons(loaded weight)} \end{bmatrix} \\ \mathcal{A} &= \text{the set of all possible values of } a, \end{aligned}$$

$R_{tt'a'}$ = the number of resources that are known at time t and before the decision at time t is made, and will be actionable with attribute vector $a' \in \mathcal{A}$ at time $t' \geq t$. Here, t is the *knowable time* and t' the *actionable time*,

$R_{tt'} = (R_{tt'a'})_{a' \in \mathcal{A}}$,

$R_t = (R_{tt'})_{t' \geq t}$.

We refer to R_t as the resource state vector. A significant issue in practice, and in this article, is the distinction between the *knowable time* (when a random variable or event becomes known) and the *actionable time*, which is when a resource is available to be acted on. If it is time 100, and an aircraft is expected to arrive at time 120 (the estimated time of arrival), then we know all the information that would have arrived by time 100 (by definition), but we cannot act on the aircraft until time 120, when it arrives.

Requirements (demands, customers) are modeled using parallel notation. We let b be the attributes of a requirement, $\mathcal{B}$ the space of potential attributes, $D_{tt'b}$ is the number of demands we know about at time t with attribute b, which are available to be moved at time t'. The demands are captured by the vector D_t. For our problem, once a demand is served it vanishes from the system, but unserved demands are held to a later time period.

For our purposes, we define the state variable $S_t = (R_t, D_t)$ to be the *physical state*. This is very common in operations research, but other communities will interpret S_t to be the information state at time t, or more generally as the state of knowledge.

Exogenous information. Exogenous information represents any changes to our state variable due to processes outside of our control. These are represented using:

W_t = the exogenous information becoming available between $t-1$ and t, such as new customer arrivals, travel delays and equipment failures,

ω = a sample path, where $W_1(\omega), W_2(\omega), \ldots,$ is a sample realization of the exogenous information,

Ω = set of sample paths,

$\hat{R}_{tt'a'}$ = the number of new resources that first become known between $t-1$ and t, and will first become actionable with attribute vector $a' \in \mathcal{A}$ at time $t' \geq t$,

$\hat{R}_{tt'} = (\hat{R}_{tt'a'})_{a' \in \mathcal{A}}$,

$\hat{R}_t = (\hat{R}_{tt'})_{t' \geq t}$.

Similarly, we let $\hat{D}_{tb}$ be the new customer demands that arrive between $t-1$ and t, with $\hat{D}_t$ being the vector of customer demands. We would let the exogenous information be $W_t = (\hat{R}_t, \hat{D}_t)$, and $W_t(\omega)$ would represent a sample realization of $\hat{R}_t$ and $\hat{D}_t$. We note that the stochastic programming community (represented by the model in Section 2.4) refers to a scenario, whereas the simulation community refers to a sample path ω (or equivalently, the sample realization of the information).

Decisions. Useful notation for modeling the types of resource allocation problems that arise in freight transportation is to define:

$\mathcal{D}$ = the set of types of decisions that can be applied to aircraft (such as move to another location, repair, reconfigure),

x_{tad} = the number of times that we apply a decision $d \in \mathcal{D}$ to an aircraft with attribute vector $a \in \mathcal{A}$ at time t,

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

For the moment, we are going to leave open precisely how we make a decision, but central to this article is the information available when we are making a decision. We model this using:

I_t = the data used to make a decision,

$X_t^\pi(I_t)$ = the function that returns decision x_t, given the information I_t. We assume that this is a member of a class of functions (policies), where we represent a member of this class using $\pi \in \Pi$.

It is very common in the dynamic programming community to assume that the state S_t, captures all the information needed to make a decision. While this representation is very compact, it hides the specifics of the information being used to make a decision. For example, we might have a forecast model $f_{tt'} = \theta_{t0} + \theta_{t1}(t' - t)$, which allows us to forecast some quantity (e.g. a demand) made at time t (using the parameter vector $\theta_t = (\theta_{t0}, \theta_{t1})$), which is known at time t. In theory, we could think of the model $f_{tt'}$ as part of our state variable (this would be consistent if we used S_t to be the state of knowledge). We are going to view the forecast model as a source of information that allows us to compute a forecast of future demand, but we are going to distinguish between the model (part of our state of knowledge) and the forecast itself. We then have to decide if we want to use the forecast when we make a decision. Many companies go through precisely this thought process when they hire a consulting firm to develop a set of forecast models. If we have such a model, we then might compute a forecast $D_{tt'} = \theta_{t0} + \theta_{t1}(t' - t)$, using the parameter vector θ_t, and then make $D_{tt'}$ part of our information vector I_t. It is our position that $D_{tt'}$ is not part of the state variable, but it is information that can be used to make a decision.

Transition function. To streamline the presentation, we are going to use a simple transition function that describes how the system evolves over time. We represent this using:

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}).$$

This function captures the effect of decisions (moving aircraft from one location to another) and information (failures, delays) on the supply of aircraft in the future, and the demands. Later, we need to specifically capture the evolution of the aircraft, captured by the resource vector, R_t. We can capture this in a similarly general way by writing:

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

In a math programming model, this is normally written as a set of linear equations. For example, we might write a stochastic version of Equation (2) describing the evolution of the resource vector for aircraft as:

$$R_{t+1} = \Delta_t x_t + \hat{R}_{t+1}, \tag{9}$$

where Δ_t is an incidence matrix that captures the results of the decisions, x_t. $\hat{R}_{t+1}$ represents exogenous (stochastic) changes to the resource vector. This could represent new aircraft arriving to the system, failed/destroyed aircraft leaving the system, and random changes to the status of an aircraft.

Evaluating the solution. To make the link with optimization, we assume there is a cost function (we can use contributions if we are maximizing) that provides a measure of the quality of a decision. This is defined using:

c_{tad} = the contribution of applying decision d to an aircraft with attribute $a \in \mathcal{A}^A$ at t,

c_t = $(c_{tad})_{a\in\mathcal{A}^A, d\in\mathcal{D}_a^A}$,

$C_t(x_t)$ = the total contribution due to x_t in time period t.

It is important to recognize that we cannot assume that the solution that provides the highest contribution always provides the most acceptable answers. Freight transportation problems are simply too complex to be captured by a single objective function.

The overall optimization problem is to find the policy π that maximizes the total contributions over all the time periods. This can be represented as:

$$\max_{\pi\in\Pi} E\left[\sum_{t\in\mathcal{T}} C_t(X_t^\pi(I_t))\right]. \tag{10}$$

Our challenge now is to define the decision function. The classical strategy is to find a function (policy) that solves the optimization problem. Our perspective is that we want to find a function that most accurately mimics the real system. In particular, we want to design decision functions that use the information that is actually available.

4. A SPECTRUM OF DECISION FUNCTIONS

To build a decision function, we have to specify the information that is available to us when we make a decision. There are four fundamental classes of information that we may use in making a decision:

—The physical state. This is our current measurement of the status of all the physical resources that we are managing.

—Forecasts of exogenous information processes. Forecasts of future exogenous information. We can use traditional point forecasts (producing deterministic models) or distributions, giving us stochastic programs such as that illustrated in Section 2.4.

—Forecasts of the impact of decisions on the future. These are functions that capture the impact on the future, of decisions made at one point in time.

—Forecasts of decisions. Patterns of behavior derived from a source of expert knowledge. These represent a form of forecast of decisions at some level of aggregation.

For notational simplicity, we let R_t represent the physical state (the state of all the resources we are managing), although it would also include customer demands, and any information about the state of the network (which might be changing dynamically). We let Ω be a forecast of future exogenous information, where Ω may be a set of future realizations (providing the basis for a stochastic model), or a single element (the point forecast). We let V_t represent the function that captures the future costs and rewards, and we let ρ be a vector representing the likelihood of making particular types of decisions.

The design of the four classes of information reflects differences in both the source of the information, and what the information is telling us (which also affects how it is used). The physical state comes from measurements of the system as it now exists. Forecasts come from forecast models, which are estimated from historical data. Forecasts project information that has not yet arrived, whereas the physical state is a measurement of the state of the system as it now exists. The values V_t, are derived from dynamic programming algorithms, and quantify (typically in some unit of currency) costs or rewards that might be received in the future. Forecasts of decisions are based on expert knowledge or past history, and are given in the same units as decisions, although these are typically at a more aggregate level.

For each information set, there are different classes of decision functions (policies). Let Π be the set of all possible decision functions that may be specified. These can be divided into three broad classes: rule-based, cost-based, and hybrid (which combine both rule-based and cost-based). These are described as follows:

Rule-based policies:

Π^{RB} = The class of rule-based policies (missing information on costs).

Cost-based policies:

Π^{MP} = The class of myopic cost-based policies (includes cost information).

Π^{RH} = The class of rolling horizon policies (includes forecasted information). In the case of deterministic future events, we get a classical deterministic, rolling horizon policy, which is widely used in dynamic settings.

Π^{ADP} = The class of approximate dynamic programming policies. We use a functional approximation to capture the impact of decisions at one point in time on the rest of the system.

Hybrid policies:

$\Pi^{EK} =$	The class of policies that use expert knowledge. This is represented using low-dimensional patterns expressed in the vector ρ. Policies in Π^{EK} combine rules (in the form of low-dimensional patterns) and costs, and therefore represent a hybrid policy.

In the following sections, we describe the information content of different policies. The following shorthand notation is used:

RB =	Rule-based (the policy uses a rule rather than a cost function). All policies that are not rule-based use an objective function.
MP =	Myopic policy which uses only information that is known and actionable now.
R =	A single requirement.
RL =	A list of requirements.
A =	A single aircraft.
AL =	A list of aircraft.
KNAN =	Known now, actionable now: policies that use information about only those resources (aircraft and requirements) that are actionable now.
KNAF =	Known now, actionable future: policies that use information about resources that are actionable in the future.
RH =	Rolling horizon policy, which uses forecasts of activities that might happen in the future.
ADP =	Approximate dynamic programming: refers to policies that use an approximation of the value of resources (in particular, aircraft) in the future.
EK =	Expert knowledge: policies that use patterns to guide behavior.

These abbreviations are used to specify the information content of a policy. The remainder of the section describes the spectrum of policies that are used in the optimizing-simulator.

4.1 Rule-Based Policies

Our rule-based policy is denoted (RB:R-A) (rule-based, one requirement, and one aircraft). In the Time-Phased Force Deployment Document (TPFDD), we pick the first available requirement and check whether the aircraft that becomes available the earliest can deliver the requirement. In this feasibility check, we examine whether the aircraft can handle the size of the cargo in the requirement, as well as whether the en route and destination airbases can accommodate the aircraft. If the aircraft cannot deliver the requirement, we check the second available aircraft, and so on, until we find an aircraft that can deliver this requirement. If it can, we upload the requirement to that aircraft, move that aircraft through a route and update the remaining weight of that requirement. After the first requirement is finished, we move to the second requirement. We continue this procedure for the requirement in the TPFDD until we finish all the requirements. See Figure 2 for an outline of this policy.

```
Step 1: Pick the first available remaining requirement in the TPFDD.
    Step 2: Pick the first available remaining aircraft.
        Step 3: Do the following feasibility check:
            Can the aircraft handle the requirement?
            Can the en route and destination airbases accommodate
            the aircraft?
        Step 4: If the answers are all yes in Step 3, deliver the
        requirement by that aircraft through a route chosen from
        the route file and update the remaining weight of
        that requirement.
    Step 5: If that requirement is not finished, go to Step 2.
Step 6: If there are remaining requirements, go to Step 1.
```

Fig. 2. Policy (RB:R-A).

This is a rule-based policy—it does not use a cost function to make the decision. We use the information set $I_t = R_{tt}$, in policy (RB:R-A), that is, only actionable resources at time t are used for the decision at time t.

4.2 Myopic Cost-Based Policies

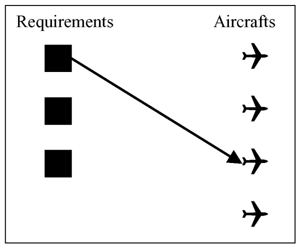
In this section, we describe a series of myopic, cost-based policies that differ in terms of how many requirements and aircraft are considered at the same time.

Policy (MP:R-AL). In policy (MP:R-AL) (myopic policy, one requirement and a list of aircraft), we choose the first requirement that needs to be moved, and then create a list of potential aircraft that might be used to move the requirement. Now we have a decision vector instead of a scalar (as occurred in our rule-based system). As a result, we now need a cost function to identify the best out of a set of decisions. Given a cost function, finding the best aircraft out of a set is a trivial sort. However, developing a cost function that captures the often unstated behaviors of a rule-based policy can be a surprisingly difficult task.

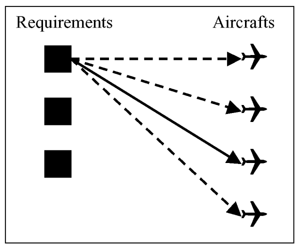
There are two flavors of policy (MP:R-AL): known now, and actionable now (MP:R-AL/KNAN), as well as known now, actionable in the future (MP:R-AL/KNAF). In the policy (MP:R-AL/KNAN), the information set $I_t = (R_{tt}, C_t)$ is used and in the policy (MP:R-AL/KNAF), the information set $I_t = ((R_{tt'})_{t' \geq t}, C_t) = (R_t, C_t)$ is used. We explicitly include the costs as part of the information set. Solving the problems requires that we sort the aircraft by their contribution and choose the one with the highest contribution.

Policy (MP:RL-AL). This policy is a direct extension of the policy (MP:R-AL). However, now we are matching a list of requirements to a list of aircraft, which requires that we solve an optimization model instead of a simple sort. In our case, this is a small integer program (it might have hundreds or even thousands of variables, but we have found these can be solved very quickly using commercial solvers). Our experimental work has shown us that problems become much harder when they are solved over longer time horizons. We use this optimization problem to balance the needs of multiple requirements and aircraft.

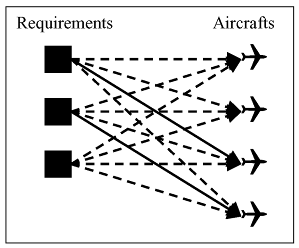
There are again two flavors of policy (MP:RL-AL): known now, and actionable now (MP:RL-AL/KNAN), as well as known now, actionable in the future



(a) Policy (RB:R-A). A rule-based policy considers only one aircraft and one requirement at a time.



(b) Policy (MP:R-AL). The policy (MP:R-AL) considers one requirement but sorts over a list of aircraft.



(c) Policy (MP:RL-AL). The policy (MP:RL-AL) works with a requirement list and an aircraft list all at the same time.

Fig. 3. Illustration of the information content of myopic policies.

(MP:RL-AL/KNAF). In the policy (MP:RL-AL/KNAN), the information set is $I_t = (R_{tt}, C_t)$, and in the policy (MP:RL-AL/KNAF), the information set is $I_t = ((R_{tt'})_{t' \geq t}, C_t) = (R_t, C_t)$. If we include aircraft and requirements that are known now but actionable in the future, then decisions that involve these resources represent *plans* that may be changed in the future.

The myopic policy $\pi \in \Pi^{MP}$ for time t is obtained by solving the following subproblem:

$$X_t^\pi(R_t) = \arg\max_{x_t \in \mathcal{X}_t} C_t(x_t).$$

Figure 3 illustrates the information considered in three examples of myopic policies. In Figure 3(a), rules are used to identify a single aircraft and a single requirement, after which the model determines if the assignment is feasible (there is no attempt to compare competing assignments). Figure 3(b) illustrates the most trivial cost-based policy, where we choose a single requirement, and then evaluate different aircraft to find the best. (The transition from the rule-based policy in 3(a) and the cost-based policy in 3(b) proved to be so difficult that the air force stayed with a rule-based policy when they undertook a complete rewrite of their simulator.) Figure 3(c) considers multiple aircraft and requirements, introducing the need for an optimization algorithm to determine the best assignment.

4.3 Rolling Horizon Policy

Our next step is to bring into play forecasted activities. A rolling horizon policy considers not only all aircraft and requirements that are known now (and possibly actionable in the future), but also forecasts of what might become known, such as: a new requirement will be in the system in two days. In the rolling horizon policy, we use information regarding states and costs arising during the planning horizon: $I_t = \{(R_{t't''})_{t'' \geq t'}, C_{t'} | t', t'' \in \mathcal{T}_t^{ph}\}$. The structure of the decision function is typically the same as with a myopic policy (but with a larger set of resources). However, we generally require the information set I_t to be treated deterministically, for practical reasons. As a result, all forecasted information has to be modeled using point estimates. If we use more than one outcome, we would be solving sequences of stochastic programs. We may present the subproblem for time t under policy $\pi \in \Pi^{RH}$:

$$X_t^\pi((R_{t'})_{t' \in \mathcal{T}_t^{ph}}) = \arg\max \sum_{t' \in \mathcal{T}_t^{ph}} C_{tt'}(x_{tt'}).$$

4.4 Approximate Dynamic Programming Policy

Dynamic programming is a technique that is used to optimize over time. If we have a deterministic problem, we would wish to solve:

$$\max_{(x_t)_{t=0}^T} \sum_{t=0}^T c_t x_t, \tag{11}$$

subject to the types of constraints used in the NRMO model in Section 2.2. This is the sort of problem that is solved using standard optimization models. Depending on the nature of the problem, we might be able to use a commercial solver, or we may have to resort to heuristics [Crainic and Gendreau 2002]. If we wish to introduce uncertainty, we would let $C(S_t, x_t)$ be the contribution we earn at time t if we are in state S_t, and make decision x_t. Since the state is random, we have to find a function (or policy), $X_t^\pi(S_t)$, that solves:

$$\max_{\pi \in \Pi} \mathbb{E}\left\{\sum_{t=0}^T C(S_t, X^\pi(S_t))\right\}. \tag{12}$$

It is well known [Puterman 1994] that the optimal policy satisfies Bellman's equation, given by:

$$V_t(S_t) \;=\; \max_{x_t \in \mathcal{X}_t} \left\{C(S_t, x_t) + \mathbb{E}\left[V_{t+1}(S_{t+1})|S_t\right]\right\}, \tag{13}$$

where $\mathcal{X}_t$ is the feasible region for time period t, and $V_t(S_t)$ is the value of being in state S_t, and following the optimal policy until the end of the horizon. Solving this equation using the classical techniques of discrete dynamic programming is well known to be computationally intractable for problems where S_t is a vector (for our problems, S_t can be a very high-dimensional vector). A strategy that helps circumvent this is to introduce an approximate value function, which we call $\bar{V}_{t+1}(R_{t+1})$. A challenge we face is that we need a value function approximation that allows us to use math programming algorithms such as linear programming to solve (13), which introduces two issues. The first is the choice of the structure of the value function. To illustrate the concepts, we are going to use a linear approximation of the form:

$$\bar{V}_{t+1}(R_{t+1}) \;=\; \sum_{a' \in \mathcal{A}} \bar{v}_{t+1,a'} R_{t+1,a'}. \tag{14}$$

Later we argue that this is actually the right approximation for the specific issues we wish to address in this problem, but for the moment we use it simply for illustrative purposes. One advantage of a linear approximation is that it does not destroy any nice problem structure that the underlying problem may have (in freight transportation, these are often large integer programs).

The second challenge is the presence of the expectation in (13). The stochastic programming model (SSDM) in Section 2.4 represents an instance where the expectation is handled by solving over a set of scenarios, but this worked because of the highly structured nature of the problem. In most applications, the number of scenarios grows exponentially in multiperiod applications.

This problem can be circumvented by first introducing the idea of the post-decision state variable, and then choosing value function approximations that are suited to the application. To illustrate, we define the post-decision resource vector by rewriting the transition Equation (9) in two steps,

$$R_t^x = \Delta_t x_t, \tag{15}$$

$$R_{t+1} = R_t^x + \hat{R}_{t+1}. \tag{16}$$

Here, R_t^x is the resource state vector resulting directly from making a decision, x_t. Thus, if we send an aircraft from Chicago to Berlin at time $t = 10$, then at time $t = 10$, this is an aircraft that will arrive in Berlin (but it is still in Chicago). $\hat{R}_{t+1}$ is the new information we will learn between time t and $t+1$ (when we make our next decision), which can include travel delays, equipment failures, and exogenous changes to the fleet.

Next, we break Bellman's Equation into two steps:

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

$$V_t^x(R_t^x) = \mathbb{E} V_{t+1}(R_{t+1}),$$

where R_{t+1} is a random variable at time t, given by Equation (16). As a rule, we cannot compute $V_t^x(R_t^x)$ exactly, so we replace it with an approximation, $\bar{V}_t(R_t^x)$. For our study, a linear value function such as (14) was appropriate, since it allowed us to capture the behavior that dispatchers did not want to send a particular type of aircraft into a region. Estimating these slopes is especially easy. Approximate dynamic programming works by stepping forward in time. In iteration n, we would follow the sample path represented by ω^n, making decisions using the value function from the previous iteration, given by $\bar{V}_t^{n-1}(R_t^x)$. If we have R_{ta}^n aircraft with attribute a, we would find x_t subject to, among other constraints, the flow conservation constraint:

$$\sum_{d \in \mathcal{D}} x_{tad} = R_{ta}^n.$$

Let $\hat{v}_{ta}^n$ be the dual variable for the flow conservation constraint. We could then use this dual to update the value function around the previous post-decision state variable, which is to say:

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

where α is a stepsize between 0 and 1.

We have used linear approximations as an illustration, but for this project, it was actually the correct functional form given what we were trying to achieve. Consider the example of a requirement that has to move from England to Australia using either a C-17 or a C-5B, two types of cargo aircraft. A problem is that the route from England to Australia involves passing through airbases that are not prepared to repair a C-5B if it breaks down, which might happen with a 10 to 20 percent probability. When a breakdown occurs, additional costs are incurred to complete the repair, which also delays the aircraft, possibly producing a late delivery, and penalties for a late delivery. Furthermore, the amount of delay, and the cost of the breakdown, can depend on whether there are other comparable aircraft present at those airbases at the time. These costs depend purely on the type of aircraft, and not the quantity, which means that a linear architecture is perfect. For different types of questions, other architectures may be more appropriate (see, for example, Tsitsiklis and Van Roy [1996]; Bertsekas and Tsitsiklis [1996]; Judd [1998]; and Powell [2007]).

Now that we have a value function approximation, take a look at the information being used to make a decision. Not only do we use resources R_t, and demands D_t, (what we have been calling our state variable), we are also using the value function approximation, $\bar{V}_t(R_t^x)$ (in the form of the slopes $(\bar{v}_t)$). There are very few in the operations research community who would view $\bar{v}_t$ as part of the state variable, but it is clearly a piece of information that we are now using to make a decision. If we use approximate dynamic programming, we obtain this information, but this is a choice. In their current simulator (AMOS), the mobility command makes an explicit choice not to use this information, presumably because it does not improve the accuracy of their model.

4.5 Expert Knowledge

All mathematical models require some level of simplification, often because of a simple lack of information. As a result, a solution may be optimal but incorrect in the eyes of a knowledgeable expert. In effect, the expert knows how the model should behave, reflecting information that is not available in the model. We represent expert knowledge in the form of low-dimensional patterns, such as "avoid sending C-5Bs on routes through India." Simulation models easily capture this sort of knowledge within their rules, but typically require that the rules be stated as hard constraints, as in "never send C-5Bs on routes through India."

Following Marar and Powell [2002], we define:

$\bar{a}$ = an attribute vector a at some level of aggregation,

$\bar{d}$ = a type of decision at some level of aggregation,

$\rho_{\bar{a}\bar{d}}$ = the fraction of instances in which decision $\bar{d}$ should be applied to a resource with attribute vector $\bar{a}$ according to expert knowledge,

ρ = $(\rho_{\bar{a}\bar{d}})_{\bar{a},\bar{d}}$,

$\bar{\rho}_{\bar{a}\bar{d}}(x)$ = fraction of time that the decision x made by the model represents acting on resources of type $\bar{a}$ with decisions of type $\bar{d}$,

$H(\bar{\rho}(x), \rho)$ = a *pattern metric* that measures the distance between the model patterns and the exogenous patterns.

Keeping in mind that the attribute vector a can be quite detailed ("a C-5B loaded with freight headed to South Korea, arriving at time 51.2") while a decision can be the assignment of an aircraft to move a specific load of freight. By contrast, patterns are typically specified at some level of aggregation. Thus, we may be concerned about "loaded C-5Bs headed to Europe." For this reason, we index patterns by an aggregated attribute vector $\bar{a}$ ("loaded C-5B") and an aggregated decision ("moving to Europe").

The pattern metric $H(\bar{\rho}(x), \rho)$ might be written:

$$H(\bar{\rho}(x), \rho) = \sum_{\bar{a}} \sum_{\bar{d}} (\rho_{\bar{a}\bar{d}}(x) - \rho_{\bar{a}\bar{d}})^2.$$

We could incorporate patterns into the cost model as:

$$X_t^{\pi}(S_t, \theta) \ = \ \arg\max_{x_t \in \mathcal{X}_t} (C(S_t, x_t) - \theta H(\bar{\rho}(x), \rho)), \tag{17}$$

where $\theta \geq 0$ serves the role of scaling pattern deviations into a cost. When we combine a cost function with a goal of matching an exogenous pattern, it is necessary to convert the degree to which we are reaching that goal into a cost-based term. As a rule, we will never perfectly match these exogenous patterns, so θ captures the importance we place on this dimension.

We represent the information content of an expert knowledge-based decision as $I_t = \{R_t, C_t, V_t, \rho\}$.

4.6 Discussion

Mathematically, an optimizing-simulator can be represented as:

$$x_t \ \leftarrow \ X_t^{\pi}(I_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \bar{V}_t(R_t^x) - \theta H(\bar{\rho}(x), \rho) \right), \tag{18}$$

$$S_{t+1} \ \leftarrow \ S^M(S_t, x_t, \hat{R}_{t+1}), \tag{19}$$

where $S^M(\cdot)$ is the transition function computed for a sample realization of W_{t+1}. We note that the choice of W_{t+1} has to be guided by a probability law in some form. It is possible to include different elements of the objective function in order to form different decision functions in the optimizing-simulator. Obviously, if nothing were in the objective function, the optimizing-simulator would be just a simple simulation model. In the optimizing-simulator, the math programming model in (18) is much smaller than that of the optimization model, since it only solves the optimization problem for one time period at a time instead of the optimization problem (as in Equation (10) or Equation (1)–(4)) for all time periods.

In this section we have introduced a series of policies, each characterized by increasing information sets. The policies are summarized in Table I, listed in order of the information content of each policy. Research has shown that the approximate dynamic programming policy (ADP) can compete with linear programming solvers on deterministic problems. For single and multi-commodity flow problems, the results are near optimal, and they significantly outperform deterministic approximations on stochastic datasets [Godfrey and Powell 2002; Topaloglu and Powell 2002; Spivey and Powell 2004]. Since the algorithmic strategy involves repeatedly simulating the system, these results are achieved without losing the generality of simulation, but the simulation must be run iteratively. Like any cost model, the policy based on the (ADP) information set means that analysts can change the behavior of the model primarily by

Table I. Information Classes and Decision Functions for Different Policies

Policy	Information Classes	Decision Functions
Rule-based	$I_t = R_{tt}$	(RB:R-A)
Myopic cost-based, one requirement to a list of aircraft, known now and actionable now	$I_t = (R_{tt}, C_t)$	(MP:R-AL/KNAN)
Myopic cost-based, one requirement to a list of aircraft, known now and actionable in the future	$I_t = ((R_{tt'})_{t' \geq t}, C_t)$	(MP:R-AL/KNAF)
Myopic cost-based, a list of requirements to a list of aircraft, known now and actionable now	$I_t = (R_{tt}, C_t)$	(MP:RL-AL/KNAN)
Myopic cost-based, a list of requirements to a list of aircraft, known now and actionable in the future	$I_t = ((R_{tt'})_{t' \geq t}, C_t)$	(MP:RL-AL/KNAF)
Rolling horizon	$I_t = \{(R_{t't''})_{t'' \geq t'}, C_{t'}, t', t'' \in \mathcal{T}_t^{ph}\}$	(RH)
Approximate dynamic programming	$I_t = \{(R_{tt'})_{t' \geq t}, C_t, V_t\}$	(ADP)
Expert knowledge	$I_t = \{(R_{tt'})_{t' \geq t}, C_t, V_t, \rho\}$	(EK)

changing costs. The final information set, ρ (used in the EK policy), allows us to manipulate the behavior of the model by changing the exogenous patterns. Needless to say, the imposition of exogenous patterns will not, in general, improve the results of the model as measured by the cost function (or even other statistics such as throughput). However, it will reduce the real costs if the expert specifies reasonable patterns.

The optimizing-simulator framework makes it possible to optimize (by which we mean use intelligent decisions) problems that are much larger and more complex than can be tackled using traditional optimization frameworks (deterministic or stochastic). This optimizing behavior is handled through the framework of approximate dynamic programming, which has a rigorous theoretical foundation. An advantage of ADP is that it never attempts to optimize large problems over all time periods at once, as is done with traditional linear programming models such as NRMO. But it does require that we step through the entire problem iteratively. We have found that the application of ADP to problems in transportation and logistics (see also Powell and Topaloglu [2005] and Simao et al. [2008]) can provide a high quality solution with as little as 50 iterations, but sometimes requires several hundred iterations. When the fleet size is decreased relative to the number of demands being served, more iterations are required. Of course, if we choose a myopic policy (ignoring the impact of decision on the future), then we may only need a single iteration.

It is useful to compare the features of the optimizing simulator to the other models that have been developed for this problem class: NRMO, AMOS, and SSDM. Table II compares each method along various dimensions in terms of their ability to model different characteristics of the problem, rather than the algorithm. The primary distinguishing feature of these models is how they have captured the flow of information, but they also differ in areas such as model flexibility, and the responsiveness to changes in input data. The optimizing simulator (O-S) representation can produce a linear programming model such as

Table II. Characteristics of NRMO, AMOS, SSDM and O-S Models

Model	NRMO	AMOS	SSDM	O-S
Category	Large-scale linear programming	Simulation	Multi-stage stochastic programming	Optimizing-simulator
Information processes	Requires knowing all information within $\mathcal{T}$ at time 0. Cannot distinguish between knowable time t and actionable time $t' \geq t$.	Assumes actionable time equals knowable time. May, but doesn't, distinguish between knowable time t and actionable time $t' \geq t$.	Actionable time equals knowable time.	General modeling of knowable and actionable time. At time t, know the information that is actionable at time $t' \geq t$.
Attribute Space	Multi-commodity flow (attribute includes aircraft type and location)	Multi-attribute (location, fuel level, maintenance)	Homogeneous ships and cargo, extendable to multiple ship types	General resource
Complexity of system dynamics	Linear systems of equations	Complex system dynamics	Simple linear systems of equations	Complex system dynamics
Information process	Deterministic	Sequential information process	Multiple scenarios	Sequential information process
Decision Selection	Cost-based	Rule-based	Cost-based	Span from rule-based to cost-based
Information Modeling	Assumes that everything is known.	Myopic, local information	Assumes knowing the probability distribution of scenarios.	General modeling of information
Model Behavior	Reacts to data changes intelligently, but not necessarily robustly across random events.	Noisy response to changes in input data	Similar to LP, but produces robust allocations.	Can react with intelligence; will display some noise characteristic of simulation; robust.
Modeling time	Physical activities in discrete time	Decisions in discrete time, physical activities in continuous time	Physical and information processes in discrete time	Decisions in discrete time, physical and information processes in continuous time

NRMO if we ignore evolving information processes, or a simulation model such as AMOS, if we explicitly model the TPFDD as an evolving information process and code the appropriate rules for making decisions. As such, the O-S representation provides a mathematical representation that spans optimization and simulation.

5. NUMERICAL EXPERIMENTS

We now undertake to demonstrate the spectrum of simulations by showing how increasing the level of information when we are making a decision improves the overall quality of the solution. We undertake these experiments using an unclassified TPFDD dataset for a military airlift problem. The problem is to manage six aircraft types (C-5A, C-5B, C-17, C-141B, KC-10A, and KC-135) to move a set of requirements of cargo and passengers between the USA and Saudi Arabia, where the total weight of the requirements is about four times the total capacities of all the aircraft. In the simulation, a typical delivery trip (pick up plus loaded movement) needs four days to complete, thus all requirements need roughly 16 days to be delivered if the capacities of all the aircraft are used. The simulation horizon is 50 days, divided into four hour time intervals. Moving a requirement involves being assigned to a route that will bring the aircraft through a series of intermediate airbases for refueling and maintenance. One of the biggest operational challenges of these aircraft is that the probability of a failure of sufficient severity to prevent a timely takeoff ranges between 10 and 25 percent. A failure can result in a delay or even require an off-loading of the freight to another aircraft. To make the model interesting, we assume that an aircraft of type C-141B (regardless of whether it is empty or loaded) has a 20 percent probability of failure, and needs five days to be repaired if it fails at an airbase in region E (airbase code names starting with E are located primarily in Northern Europe). All other aircraft types or airbases are assumed to be able to repair the failures without delay.

The TPFDD file does not capture the time when the information about a requirement becomes known. For our experiments, we assumed that requirements are known two days before they have to be moved. Aircraft, on the other hand, are either actionable now (if they are empty and on the ground) or are actionable at the end of a trip that is in progress. We assume that there are three types of costs involved in the military airlift problem: transportation costs (2 cents per mile per pound of capacity of an aircraft), aircraft repair costs (6 cents per period per pound of capacity of a disabled aircraft) and penalties for late deliveries (4 cents per period per pound of requirement delivered late).

We have such a rich family of models that it would become clumsy if we compared all the policies introduced in Section 4. To focus on the main idea of this article, we run the optimizing-simulator on the following policies: (1) rule-based, one requirement to one aircraft (RB:R-A), (2) cost-based, one requirement to a list of aircraft that are knowable now, actionable now (MP:R-AL/KNAN), (3) cost-based, a list of requirements to a list of aircraft that are knowable now, actionable now (MP:RL-AL/KNAN), (4) the same policy but with aircraft that are knowable now, actionable in the future (MP:RL-AL/KNAF), and (5) the

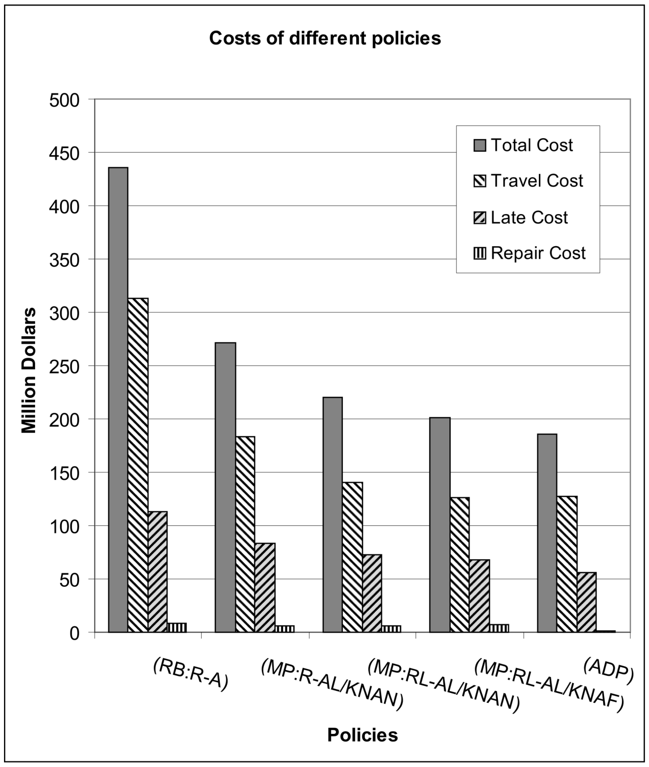


Fig. 4. Costs of different policies.

approximate dynamic programming policy (ADP). These five classes should provide improved solutions as they are added. We did not explicitly test rolling horizon procedures since this would have required generating a forecast of future events from the TPFDD. This would be straightforward in the context of a civilian application such as freight transportation where historical activities would form the basis of a forecast, but a historical record does not exist for these applications.

We use three measures of solution quality. The first is the traditional measure of the objective function. It is important to emphasize that this is an imperfect measure, since some behaviors may not be reflected in a cost function. The second measure is throughput, which is of considerable interest in the study of airlift problems. Our cost function captures throughput indirectly through costs that penalize late deliveries. Finally, when we study the use of expert knowledge, we measure the degree to which the model matches exogenously specified patterns.

Figure 4 shows the costs for each of the first five policies. Policy (RB:R-A) is rule-based, one requirement to one aircraft. Policy (MP:R-AL/KNAN) is cost-based, one requirement to a list of aircraft that are knowable now actionable now. Policy (MP:RL-AL/KNAN) is cost-based, a list of requirements to a list of aircraft that are knowable now, actionable now. Policy (MP:RL-AL/KNAF) is

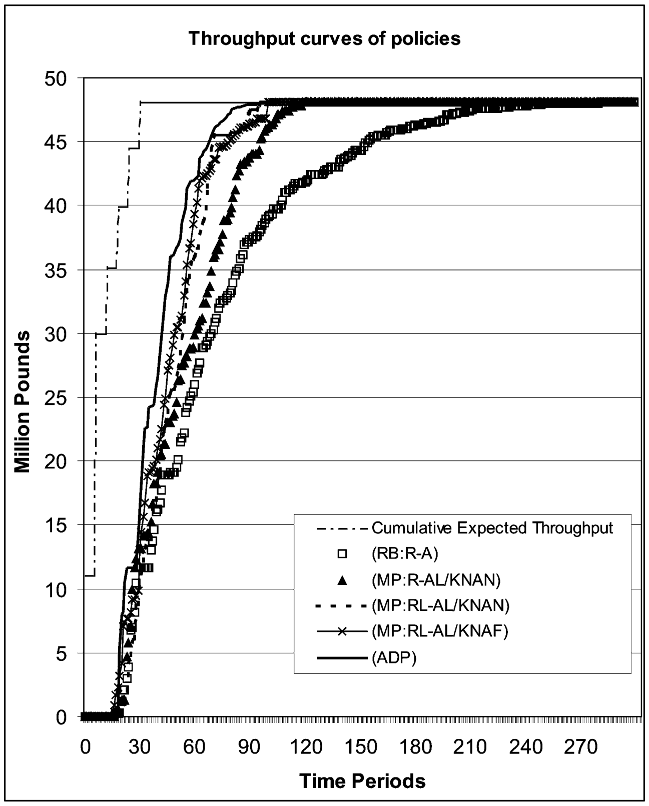


Fig. 5. Throughput curves of different policies.

cost-based, a list of requirements to a list of aircraft that are knowable now actionable in the future. Policy (ADP) is the approximate dynamic programming policy. The total cost is the sum of transportation costs, late delivery costs, and aircraft repair costs. The late delivery costs decrease steadily as the information set increases. The repair cost is significantly reduced in policy (ADP) since this policy learns from the early iterations and avoids sending aircraft to airbases that lead to longer repair times. However, the detour increases the transportation cost of policy (ADP) slightly compared to policy (MP:RL-AL/KNAF). The overall total costs are decreasing as we expected, since the information sets are increasing.

The throughput of each of the five different policies (RB:R-A), (MP:R-AL/KNAN), (MP:RL-AL/KNAN), (MP:RL-AL/KNAF), and (ADP), are plotted in Figure 5, which shows cumulative pounds delivered over the simulation. Also shown is the cumulative expected throughput curve, which represents the cumulative total tonnage that has been requested to move. The cumulative expected throughput curve assumes that every unit of demand is moved instantaneously, so this represents the best that the system can do.

The throughput also follows this sequence, from the right to the left. It is clear that the richer the information class, the faster the delivery—i.e. the closer to

Table III. Areas Between the Cumulative Expected Throughput Curve and the Throughput Curves of Different Policies

Policy	pounds * day
(RB:R-A)	472,868,381
(MP:R-AL/KNAN)	344,977,669
(MP:RL-AL/KNAN)	303,568,943
(MP:RL-AL/KNAF)	281,365,953
(ADP)	234,915,133

the left is the throughput curve. Since some of the throughput curves cross each other, we calculate the area between the expected throughput and the throughput curves of different policies and list them in Table III. These areas actually measure the lateness of the delivery of different policies. The smaller the area is, the faster the delivery is. We may see that from policy (RB:R-A) to (ADP), the areas are decreasing from 473 million to 235 million (pound Days).

We tested the inclusion of expert knowledge by using an exogenous pattern to control the percentage of C-5As and C-5Bs through region E. Starting with the best myopic policy (MP:RL-AL/KNAF), we found that C-5As and Bs went through region E 18.5 percent of the time. We then imposed a single exogenous pattern on the (attribute, decision) pair (a = C-5A or C-5B, d = region E). We then varied the exogenous pattern $\rho_{\bar{a}\bar{d}}$ from 0 to 0.5 in steps of 0.10. For example, $\rho_{\bar{a}\bar{d}} = 0.3$, indicates that as recommended by the expert, C-5As and C-5Bs should be sent through region E in 30% of the instances in which a decision is to be applied to them.

Choosing the correct value of the pattern weight, θ, (which determines the deviation of the final solution from the solution that exactly matches the exogenous pattern) is a matter of subjective judgment. If the cost function has been carefully designed, then some amount of deviation may still be acceptable. For each value of $\rho_{\bar{a}\bar{d}}$, we varied θ over the values 0, 3, 5, 10, 100, 1000.

The results are shown in Figure 6, which maps the observed pattern from the model to the expert-recommended pattern. The horizontal line corresponds to $\theta = 0$, and we also show a 45 degree dashed line representing the case that the model matches the pattern exactly. For the remaining runs, varying θ for different values of $\rho_{\bar{a}\bar{d}}$ produces a series of lines that are bounded by the no-pattern and the exact-match lines. Note that matching an exogenous pattern will typically produce a lower objective function (or higher costs). The point of matching patterns is to produce a behavior that is not captured by the objective function.

These results indicate that we can retain the ability that exists in traditional simulators to guide the behavior of the cost model using simple rules. It is important to emphasize, however, that our exogenous patterns are restricted in their complexity. A rule must be a function of the (attribute, decision) pair $(\bar{a}, \bar{d})$, which means that the rule may not reflect, for example, information about other aircraft (the cost model must pick this up).

The point of these experiments is not to conclude that one decision function is better than another, since the objective function (Figure 4) or throughput (Figure 5) represent only two measures of a solution. It is interesting that as

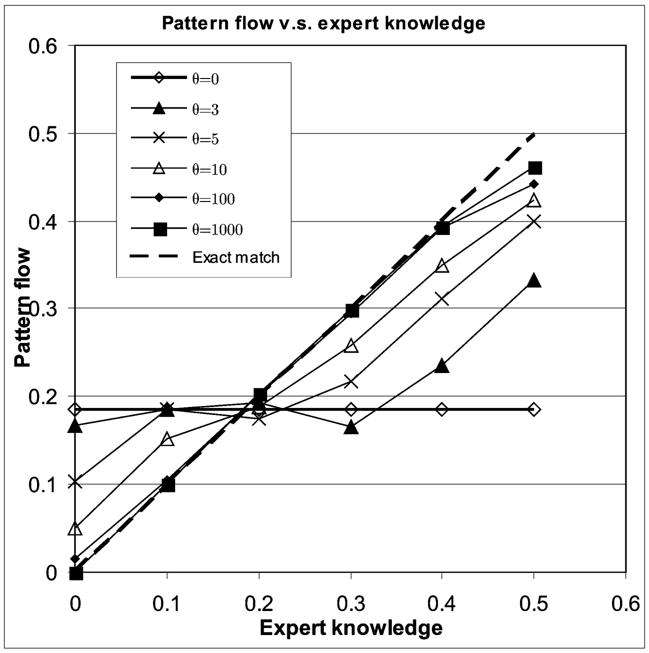


Fig. 6. Pattern match under different θ weights.

of this writing, the analysis group at the air mobility command continues to use the simplest rule-based logic, even after contracting to have their original simulation package completely rewritten.

6. CONCLUSIONS

The modeling community in transportation, and in particular in the area of airlift modeling, has traditionally been divided between simulation, which offers tremendous flexibility as well as the ability to handle uncertainty, and optimization, which offers high quality solutions but limits our ability to handle uncertainty as well as more complex dynamics. These have been viewed as competing methodologies, and the communities that promote them have little overlap.

We have shown, using the context of modeling cargo movements for military operations, that simulation and optimization can be viewed as different types of decision functions using different types of information. Decision functions can be rule-based or cost-based, and we illustrate them by showing how different types of functions can be created using four classes of information. While there is a desire, especially in the academic community, to find the best solution (or the best policy), it is often the case that in practice the goal is a model that mimics actual operations. This means not only modeling the physics of the problem (travel times, capacities and constraints), but also the way that decisions are made. A major limitation that has often been cited by the analysis group at the airlift mobility command is that optimization models are "too smart."

For the airlift problem we considered, we showed that increasing the information available to a decision-maker improves the solution quality in terms of specific metrics. However, the logic anticipates that there may be issues we are not capturing, and we show how we can design decision functions that allow an analyst to guide the model using explicit patterns of behavior. These patterns are expressed as a form of goal, rather than the hard rules that are more common in simulation models.

The academic literature has focused on finding the best decision (for deterministic problems) or the best policy (for stochastic problems). We suggest that a new line of research falls in the area of model calibration. Companies who want to use these models to answer high-level policy questions gain confidence when they feel that the model closely matches the performance of their operation. For civilian operations, historical data may be available, introducing the new challenge of designing procedures that produce the best match by manipulating the policy. This article shows that by controlling the available information, we can create a broad range of policies, including those that directly compete with optimization models. An interesting research challenge would be to create automated procedures that identify the policies that produce the closest match between model and history.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the very helpful comments of the referees.

REFERENCES

Baker, S., Morton, D., Rosenthal, R., and Williams, L. 2002. Optimizing military airlift. *Oper. Res. 50*, 4, 582–602.

Bertsekas, D. and Tsitsiklis, J. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont.

Brooke, A., Kendrick, D., and Meeraus, A. 1992. *GAMS: A User's Guide*, 2.25 ed. Duxbury Press/Wadsworth Publishing Company, Belmont.

Burke, J. F., Love, R. J., and Macal, C. M. 2004. Modelling force deployments from army installations using the transportation system capability (TRANSCAP) model: A standardized approach. *Math. Comput. Model. 39*, 6-8, 733–744.

Crainic, T. and Gendreau, M. 2002. Cooperative parallel tabu search for capacitated network design. *J. Heuristics 8*, 6, 601–627.

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. *Math. Comput. Model. 39*, 6-8, 599–616.

Dantzig, G. and Ferguson, A. 1956. The allocation of aircraft to routes: An example of linear programming under uncertain demand. *Manage. Sci. 3*, 45–73.

Ferguson, A. and Dantzig, G. B. 1955. The problem of routing aircraft—a mathematical solution. *Aeronaut. Eng. Rev. 14*, 51–55.

Fu, M. 2002. Optimization for simulation: Theory vs. practice. *INFORMS J. Comput. 14*, 3, 192–215.

Godfrey, G. and Powell, W. B. 2002. An adaptive, dynamic programming algorithm for stochastic resource allocation problems I: Single period travel times. *Transport. Sci. 36*, 1, 21–39.

Goggins, D. A. 1995. Stochastic modeling for airlift mobility. M.S. thesis, Naval Postgraduate School, Monterey.

GRANGER, J., KRISHNAMURTHY, A., AND ROBINSON, S. M. 2001. Stochastic modeling of airlift operations. In *Proceedings of the 2001 Winter Simulation Conference*, B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, Eds. 432–440.

JUDD, K. 1998. *Numerical Methods in Economics*. MIT Press.

KILLINGSWORTH, P. AND MELODY, L. J. 1997. Should C17's be deployed as theater assets?: An application of the CONOP air mobility model. Tech. Rep. rand/db-171-af/osd, Rand Corporation.

MARAR, A. AND POWELL, W. B. 2002. Using static flow patterns in time-staged resource allocation problems. Tech. Rep., Princeton University, Department of Operations Research and Financial Engineering.

MATTOCK, M. G., SCHANK, J. F., STUCKER, J. P., AND ROTHENBERG, J. 1995. New capabilities for strategic mobility analysis using mathematical programming. Tech. Rep., RAND Corporation.

MIDLER, J. L. AND WOLLMER, R. D. 1969. Stochastic programming models for airlift operations. *Nav. Res. Logist. Quart. 16*, 315–330.

MORTON, D. P., ROSENTHAL, R. E., AND LIM, T. W. 1996. Optimization modeling for airlift mobility. *Mil. Oper. Res.* 49–67.

MORTON, D. P., SALMERON, J., AND WOOD, R. K. 2002. A stochastic program for optimizing military sealift subject to attack. *Stochastic Programming e-Print Series*, http://www.speps.info.

NIEMI, A. 2000. *Stochastic Modeling for the NPS/RAND Mobility Optimization Model*. Department of Industrial Engineering, University of Wisconsin-Madison, http://ie.engr.wisc.edu/robinson/Niemi.htm.

POWELL, W. B. 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley and Sons, New York.

POWELL, W. B. AND TOPALOGLU, H. 2005. Fleet management. *Applications of Stochastic Programming*, S. Wallace and W. Ziemba, Eds. Math Programming Society—SIAM Series in Optimization, Philadelphia.

PUTERMAN, M. L. 1994. *Markov Decision Processes*. John Wiley & Sons, New York.

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. *Mil. Oper. Res. 3*, 2, 55–74.

SCHANK, J., MATTOCK, M., SUMNER, G., GREENBERG, I., AND ROTHENBERG, J. 1991. A review of strategic mobility models and analysis. Tech. Rep., RAND Corporation.

SIMAO, H. P., DAY, J., GEORGE, A. P., GIFFORD, T., NIENOW, J., AND POWELL, W. B. 2008. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transport. Sci. 43*, 2, 178–197.

SPIVEY, M. AND POWELL, W. B. 2004. The dynamic assignment problem. *Transport. Sci. 38*, 4, 399–419.

STUCKER, J. P. AND BERG, R. T. 1999. Understanding airfield capacity for airlift operations. Tech. Rep., RAND Corporation.

SWISHER, J., JACOBSON, S., AND YÜCESAN, E. 2003. Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Trans. Model. Comput. Simul. 13*, 2, 134–154.

TOPALOGLU, H. AND POWELL, W. B. 2002. Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems. Tech. Rep., Princeton University, Department of Operations Research and Financial Engineering.

TSITSIKLIS, J. N. AND VAN ROY, B. 1996. Feature-based methods for large scale dynamic programming. *Mach. Learn. 22*, 59–94.

WING, V., 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.

YOST, K. A. 1994. The thruput strategic airlift flow optimization model. Tech. Rep., Air Force Studies and Analyses Agency, The Pentagon, Washington D.C.

YOST, K. A. AND WASHBURN, A. R. 2000. The LP/POMDP marriage: Optimization with imperfect information. *Nav. Res. Logist. 47*, 8, 607–619.

Received July 2008; revised September 2008; accepted October 2008