INFORMS Journal on Computing
Vol. 18, No. 1, Winter 2006, pp. 31–42
ISSN 1091-9856 | EISSN 1526-5528 | 06 | 1801 | 0031

DOI 10.1287/ijoc.1040.0079
© 2006 INFORMS

Dynamic-Programming Approximations for Stochastic Time-Staged Integer Multicommodity-Flow Problems

Huseyin Topaloglu
School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853, USA,
topaloglu@orie.cornell.edu

Warren B. Powell
Department of Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey 08544, USA,
powell@princeton.edu

In this paper, we consider a stochastic and time-dependent version of the min-cost integer multicommodity-flow problem that arises in the dynamic resource allocation context. In this problem class, tasks arriving over time have to be covered by a set of indivisible and reusable resources of different types. The assignment of a resource to a task removes the task from the system, modifies the resource, and generates a profit. When serving a task, resources of different types can serve as substitutes of each other, possibly yielding different revenues. We propose an iterative, adaptive dynamic-programming-based methodology that makes use of linear or nonlinear approximations of the value function. Our numerical work shows that the proposed method provides high-quality solutions and is computationally attractive for large problems.

Key words: dynamic programming; networks-graphs; multicommodity; transportation
History: Accepted by William Cook, Area Editor for Design and Analysis of Algorithms; received August 2003; revised February 2004; accepted March 2004.

1. Introduction

Dynamic resource-allocation problems involve the assignment of a set of reusable resources to tasks that occur over time. The arrival process of the tasks is known only through a probability distribution. The assignment of a resource to a task produces a reward, removes the task from the system, and modifies the state of the resource. Such problems arise in many fields such as dynamic fleet management, product distribution, machine scheduling, and personnel management. In this paper, we consider dynamic resource-allocation problems in which there is substitution among resources. Different types of resources can be used to cover a task, and covering a task with different types of resources may yield different rewards.

We were confronted with this problem within the context of managing a fleet of business jets to serve customers who request to be flown between different airports. The fleet is composed of different types of jets. Due to operating costs and customer preferences, satisfying a certain customer demand with different types of jets yields different revenues. Given the high repositioning costs and substitution penalties, effective repositioning of the available jets to serve the uncertain future customer requests and using the "correct" type of jet to satisfy a demand are crucial. Clearly, the jet type that maximizes the immediate profit is very often not necessarily the best pick.

The deterministic version of this problem is the min-cost integer multicommodity-flow problem. The linear and integer versions of the min-cost multicommodity-flow problem have been studied extensively. Early survey papers by Assad (1978) and Kennington (1978) describe various solution approaches to the linear version. The integer version has seen increased attention due to its wide applicability to fields such as airline fleet assignment (Hane et al. 1995), car and container distribution in the railroad and maritime industries (Jordan and Turnquist 1983, Shah 1985, Chih 1986, Crainic et al. 1993, Holmberg et al. 1998), and dynamic fleet management (Powell et al. 1995). All of these problem instances involve solving the min-cost integer multicommodity-flow problem on a state-time network, where the state is typically the location of the physical resource being managed. However, the literature that explicitly tries to incorporate stochastic elements is not as rich except for a few instances where the stochastic elements of the problem are in the objective function (Aneja and Nair 1982, Soroush and Mirchandani 1990).

The approach we follow in this paper formulates the problem as a dynamic program and replaces the value function by a separable continuous approximation. We update and improve the approximation using samples of the random quantities. Our work here builds on prior research. Powell and Carvalho (1998) suggest the use of linear approximations for determin-

istic multistage resource-allocation problems. Godfrey and Powell (2002) introduce a separable adaptive nonlinear approximation technique for stochastic, multistage resource-allocation problems with a single resource type. This paper extends the aforementioned work in three ways to handle problems with multiple resource types. (1) Powell et al. (2002) present a new algorithm for building separable approximations of the value function and show that it is convergent for certain two-stage resource-allocation problems. This paper empirically investigates the effectiveness of this new approximation method when applied to multistage resource-allocation problems. (2) If we try to apply the approaches in Powell and Carvalho (1998) or Godfrey and Powell (2002) to a problem with multiple resource types, we have to solve a min-cost integer multicommodity-flow problem for each time period. These problems tend to get large easily with the number of possible states and resource types, and their multicommodity nature presents an unwelcome dimension of complexity. We develop a new value-function approximation method using a *mixture* of linear and nonlinear value-function approximations that requires solving sequences of min-cost network-flow problems as opposed to min-cost integer multicommodity-flow problems. (3) The test problems used in Powell and Carvalho (1998) and Godfrey and Powell (2002) arise from the dynamic fleet-management setting, and therefore exhibit a natural geographic separability. It is not very surprising that separable approximations work well in this setting. On the other hand, when we have multiple types of vehicles that can exist in the same geographical location and can compete to serve the same set of demands, it is very hard to assume that the problem decomposes by the vehicle types. In this case, the success of separable approximations is not as obvious.

General stochastic-programming approaches are not suitable for our problem class for several reasons. It is hard to store the deterministic equivalent linear program because it involves a large number of scenarios and decision variables (deterministic instances of our problem class are often difficult). Algorithms based on Benders decomposition, such as Higle and Sen (1991) and Chen and Powell (1999), have difficulty with satisfying the integrality requirements, and Powell et al. (2002) show that they may suffer from slow convergence. Recently, there has been an increasing interest in approximate dynamic programming; Bertsekas and Tsitsiklis (1996) give a structured coverage of this literature. Methods based on discrete representations of the value-function approximations are intractable for our problem class because the number of possible states is huge. As we show in this paper, linear approximations are easy to work with but are typically unstable and do not provide high solution quality. Nonlinear polynomial approximations have been proposed, but when used with the classical optimality recursion, they require explicit computation of expectations and pose complications for problems with integrality requirements.

In this paper we make the following research contributions. (1) We propose a new dynamic-programming approximation method for dynamic allocation of substitutable resources under uncertainty. Our method uses a hybrid of linear and piecewise-linear approximations of the value function. We show that with this approximation strategy, we have to solve sequences of min-cost network-flow problems, which yield integer solutions naturally. (2) Theoretically, the extension of our method to the case where the resource transformations (travel times) take more than one time period is trivial. However, in practice it may take our approach much longer to provide high-quality solutions for problems with multiperiod transfer times. We derive a key result that accelerates the performance of our approach in the presence of multiperiod transfer times. (3) We experimentally show that our method is computationally tractable and attractive. For deterministic instances of the problem, it yields solutions that are very close to the upper bound on the optimal value of the objective function. For stochastic instances, it outperforms standard deterministic rolling-horizon procedures, typically by significant margins.

In Section 2, we define the problem and introduce the notation. Section 3 describes the formulation of the problem as a dynamic program and the basic idea of approximating the value function. Section 4 discusses different ways of approximating the value function and establishes the structural properties of the problems that have to be solved for each time stage. Section 5 describes how to update the approximations using samples of the random variables and resolves difficulties arising from multiperiod resource-transformation times. Section 6 presents our numerical results.

2. Problem Formulation

In this section, we formulate our dynamic resource-allocation problem using the language of Markov decision processes. The presentation follows the jet-management problem that is the motivating application for this work, but the basic model is applicable in many other contexts.

We have a fleet of business jets of different types. At each decision epoch, a certain number of customers call in, each requesting to be flown from a certain origin to a destination. The customers tend to call in at the last minute and very little advance information about the future requests is available. We are required to serve every customer demand. However, if there

are not enough in-house jets, the unsatisfied customer demands are served by using chartered jets. To handle this, we assume that the unsatisfied demands are lost (served with a chartered jet), and we take the profit from serving a certain demand to be the incremental profit from serving the demand with an in-house jet over the profit from serving the demand with a chartered jet.

Our initial formulation assumes that all flights take a single time period and all customers have the same preferences for different jet types. We show how to relax these assumptions later in this section and present its implications on our solution methodology in Section 5.3. For notational convenience, we also assume that a demand at a certain location can be served only by a jet at the same location. For the rest of the paper, we adopt the standard fleet-management vocabulary and refer to jets as "vehicles" and serving a demand as "moving loaded." We first define the following.

$\mathcal{T}$ = Set of time periods over which the demands occur, $\{1, \ldots, T\}$.
$\mathcal{I}$ = Set of locations.
$\mathcal{K}$ = Set of vehicle types.
D_{ijt} = (Random) number of demands that need to be carried from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$. We assume that D_{ijt} and $D_{i'j't'}$ are independent for $t \neq t'$.

The decisions we can apply on the vehicles and the relevant costs are the following.

x^{ke}_{ijt} = Number of vehicles of type $k \in \mathcal{K}$ moving empty from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$.
x^{kl}_{ijt} = Number of vehicles of type $k \in \mathcal{K}$ moving loaded from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$.
c^{ke}_{ijt} = Cost of an empty movement of a vehicle of type $k \in \mathcal{K}$ from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$.
c^{kl}_{ijt} = Net profit from a loaded movement of a vehicle of type $k \in \mathcal{K}$ from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$.
R^k_{it} = Number of vehicles of type $k \in \mathcal{K}$ at location $i \in \mathcal{I}$ at time period $t \in \mathcal{T}$.

The deterministic version of the problem we are interested in can be written as

$$\max \sum_{t\in\mathcal{T}} \sum_{i,\, j\in\mathcal{I}} \sum_{k\in\mathcal{K}} \left(-c^{ke}_{ijt} x^{ke}_{ijt} + c^{kl}_{ijt} x^{kl}_{ijt}\right) \tag{1}$$

$$\text{subject to} \quad \sum_{j\in\mathcal{I}} \left(x^{ke}_{ij1} + x^{kl}_{ij1}\right) = R^k_{i1} \quad i \in \mathcal{I},\ k \in \mathcal{K}$$

$$-\sum_{j\in\mathcal{I}} \left(x^{ke}_{ji,\,t-1} + x^{kl}_{ji,\,t-1}\right) + \sum_{j\in\mathcal{I}} \left(x^{ke}_{ijt} + x^{kl}_{ijt}\right) = 0 \quad i \in \mathcal{I},\ k \in \mathcal{K},\ t \in \mathcal{T}\setminus\{1\}$$

$$\sum_{k\in\mathcal{K}} x^{kl}_{ijt} \le D_{ijt} \quad i, j \in \mathcal{I},\ t \in \mathcal{T}$$

$$x^{ke}_{ijt},\ x^{kl}_{ijt} \in \mathbb{Z}_+ \quad i, j \in \mathcal{I},\ k \in \mathcal{K},\ t \in \mathcal{T},$$

which is a special case of the min-cost integer multicommodity-flow problem.

We denote the vectors $[D_{ijt}]_{i,\, j\in\mathcal{I}}$, $[x^{ke}_{ijt}]_{i,\, j\in\mathcal{I},\, k\in\mathcal{K}}$, $[x^{kl}_{ijt}]_{i,\, j\in\mathcal{I},\, k\in\mathcal{K}}$, $[c^{ke}_{ijt}]_{i,\, j\in\mathcal{I},\, k\in\mathcal{K}}$, $[c^{kl}_{ijt}]_{i,\, j\in\mathcal{I},\, k\in\mathcal{K}}$, and $[R^k_{it}]_{i\in\mathcal{I},\, k\in\mathcal{K}}$ by D_t, x^e_t, x^l_t, c^e_t, c^l_t, and R_t, and we denote $[x^e_t \mid x^l_t]$ and $[-c^e_t \mid c^l_t]$ by x_t and c_t, respectively. Then we can use R_t as the state variable to formulate the stochastic version of the problem as a dynamic program. Assuming all decisions take a single time period to implement, the dynamics of the system at time t are given by

$$R^k_{j,\,t+1} = \sum_{i\in\mathcal{I}} \left(x^{ke}_{ijt} + x^{kl}_{ijt}\right) \quad j \in \mathcal{I},\ k \in \mathcal{K}. \tag{2}$$

Given R_t and D_t, the set of feasible decisions at time t is

$$\mathcal{X}(R_t, D_t) = \Bigg\{ x_t :\ \sum_{j\in\mathcal{I}} \left(x^{ke}_{ijt} + x^{kl}_{ijt}\right) = R^k_{it} \quad i \in \mathcal{I},\ k \in \mathcal{K} \tag{3}$$

$$\sum_{k\in\mathcal{K}} x^{kl}_{ijt} \le D_{ijt} \quad i, j \in \mathcal{I} \tag{4}$$

$$x^{ke}_{ijt},\ x^{kl}_{ijt} \in \mathbb{Z}_+ \quad i, j \in \mathcal{I},\ k \in \mathcal{K} \Bigg\}. \tag{5}$$

We also set

$$\mathcal{Y}(R_t, D_t) = \Bigg\{ (x_t, R_{t+1}) :\ R^k_{j,\,t+1} = \sum_{i\in\mathcal{I}} \left(x^{ke}_{ijt} + x^{kl}_{ijt}\right) \quad j \in \mathcal{I},\ k \in \mathcal{K},\ x_t \in \mathcal{X}(R_t, D_t) \Bigg\}.$$

Thus, $(x_t, R_{t+1}) \in \mathcal{Y}(R_t, D_t)$ means that x_t is a feasible decision when the state of the system is R_t and demand outcome is D_t, and applying the decision x_t on the state vector R_t generates the state vector R_{t+1} for the next time period.

We are interested in finding a policy that maximizes the expected profit over all time periods. Because the random variables D_{ijt} and $D_{i'j't'}$ are independent for $t \neq t'$, it can be shown that the optimal policy is Markovian (it depends on the history of the system only through the current state) and deterministic. Markovian deterministic policies define one decision function for each time period t that maps the state of the system (R_t) and the outcome of the random variables (D_t) at time period t to a set of decisions. Denoting the sequence of decision functions corresponding to policy π by $\{X^\pi_t :\ t \in \mathcal{T}\}$, we are interested in finding the policy π^* that maximizes

$$\mathbb{E}\left\{ \sum_{t\in\mathcal{T}} c_t X^\pi_t(R_t, D_t) \,\middle|\, R_1 \right\}.$$

By the principal of optimality (Bellman 1957), we can find the optimal policy by solving

$$V_t(R_t) = \mathbb{E}\left\{ \max_{(x_t,\, R_{t+1})\in\mathcal{Y}(R_t,\, D_t)} c_t x_t + V_{t+1}(R_{t+1}) \,\middle|\, R_t \right\}. \tag{6}$$

Remark. We can extend our formulation to allow customers to have different preferences for different jet types. For example, high-end customers may aggressively demand a certain type of jet and substitutions may be costly, whereas low-end customers may be willing to switch to any type of jet without much penalty. For this purpose we define the following.

$\mathcal{D}$ = Set of customers.

D^d_{ijt} = (Random) number of demands from customer $d \in \mathcal{D}$ that need to be carried from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$.

x^{kdl}_{ijt} = Number of vehicles of type $k \in \mathcal{K}$ moving loaded from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$ and serving a demand from customer $d \in \mathcal{D}$.

c^{kdl}_{ijt} = Net profit from a loaded movement of a vehicle of type $k \in \mathcal{K}$ from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$ at time period $t \in \mathcal{T}$ that serves a demand from customer $d \in \mathcal{D}$.

With these definitions, equations (2), (3), and (4) can be modified as

$$R^k_{j,t+1} = \sum_{i\in\mathcal{I}}\left(x^{ke}_{ijt} + \sum_{d\in\mathcal{D}} x^{kdl}_{ijt}\right) \quad j \in \mathcal{I},\ k \in \mathcal{K}$$

$$\sum_{j\in\mathcal{I}}\left(x^{ke}_{ijt} + \sum_{d\in\mathcal{D}} x^{kdl}_{ijt}\right) = R^k_{it} \quad i \in \mathcal{I},\ k \in \mathcal{K}$$

$$\sum_{k\in\mathcal{K}} x^{kdl}_{ijt} \le D^d_{ijt} \quad i, j \in \mathcal{I},\ d \in \mathcal{D}.$$

Remark. We can easily extend our formulation to cover cases where there are multiperiod travel times by defining the following.

$R^k_{it't}$ = Number of vehicles of type $k \in \mathcal{K}$ that are inbound to location $i \in \mathcal{I}$ at time period $t \in \mathcal{T}$ and will arrive at location $i \in \mathcal{I}$ at time period $t' \in \mathcal{T}$.

τ_{ij} = Travel time from location $i \in \mathcal{I}$ to $j \in \mathcal{I}$.

The system dynamics at time t are now described by

$$R^k_{jt',t+1} = \sum_{i\in\mathcal{I}}\left(\mathbf{1}_{\tau_{ij}}(t'-t)x^{ke}_{ijt} + \mathbf{1}_{\tau_{ij}}(t'-t)x^{kl}_{ijt}\right) + R^k_{jt't} \quad j \in \mathcal{I},\ k \in \mathcal{K},\ t' > t,$$

where we use

$$\mathbf{1}_y(x) = \begin{cases} 1 & x = y, \\ 0 & \text{otherwise.} \end{cases}$$

The set of feasible decisions is given by

$$\mathcal{X}(R_t, D_t) = \left\{x_t:\ \sum_{j\in\mathcal{I}}\left(x^{ke}_{ijt} + x^{kl}_{ijt}\right) = R^k_{itt} \quad i \in \mathcal{I},\ k \in \mathcal{K},\ (4),\ (5)\right\}.$$

Having very long travel times has a tendency to reduce the performance of our solution approach. We address this issue in Section 5.3.

For clarity of presentation in the rest of the paper, unless we note otherwise, we assume that we have a single customer type and the travel times are one time period. In Section 6, where we present our experimental results, we drop these assumptions.

3. Solution Methodology

For almost all problem instances of practical significance, solving (6) is intractable for three reasons: (1) The number of possible values of R_t may be very large and $V_t(R_t)$ has to be evaluated for all possible values of R_t. (2) Computing the expectation may be intractable. (3) Unless V_{t+1} has a special structure, solving the maximization problem may not be easy. To overcome these difficulties, we propose a methodology based on Monte Carlo samples of the random quantities (which avoids computing expectations) and that uses specially structured approximations of the value function (which makes solving the maximization problem easy).

We let $V_t(R_t) = \mathbb{E}\{V_t(R_t, D_t) \mid R_t\}$, where

$$V_t(R_t, D_t) = \max_{(x_t, R_{t+1})\in\mathcal{Y}(R_t, D_t)} c_t x_t + V_{t+1}(R_{t+1}). \quad (7)$$

Now, we replace the value function V_{t+1} with a suitable approximation, denoted $\hat{V}_{t+1}$, and solve the following problem for *one* Monte Carlo sample of D_t (denoted by $\hat{D}_t$):

$$\tilde{V}_t(R_t, \hat{D}_t) = \max_{(x_t, R_{t+1})\in\mathcal{Y}(R_t, \hat{D}_t)} c_t x_t + \hat{V}_{t+1}(R_{t+1}). \quad (8)$$

We refer to problem (8) for a given value of R_t and $\hat{D}_t$ as a *subproblem* for time period t. Starting with a set of value-function approximations and an initial state vector, we sequentially solve one subproblem for each $t \in \mathcal{T}$, using one sample of D_t. Our challenge is to devise a method for using the information obtained while solving (8) to update and improve the value-function approximation $\hat{V}_t$. After the updating procedure, we obtain a new set of value-function approximations. Then, we solve all the subproblems using the new value-function approximations and new sample realizations.

If $\hat{D}^n_t$ is the demand sample, R^n_t is the state variable, and $\hat{V}^n_t$ is the value-function approximation in the subproblem for time period t at iteration n, we need to design an updating scheme that we may, for the moment, represent using $\hat{V}^{n+1}_t \leftarrow \mathcal{U}(\hat{V}^n_t, \hat{D}^n_t, R^n_t)$. This idea is presented in Figure 1. In the following two sections, we describe how to construct and update the value-function approximations.

4. Approximating the Value Function

V_t is a concave function on the integer lattice (Haneveld and van der Vlerk 1999). That is, for all $R^0_t, R^1_t, R^2_t \in \mathbb{Z}^{|\mathcal{I}||\mathcal{K}|}_+$ and $\alpha \in (0, 1)$ such that $R^0_t = \alpha R^1_t +$

Step 0 Initialization: choose an approximation $\hat{V}_t$ for V_t for all $t \in \mathcal{T}$. Set iteration counter $n = 1$.

Step 1 Forward Pass:

Step 1.0 Initialize the forward pass: initialize R_1 to the initial state of the vehicles. Obtain a sample realization of $\{D_t : t \in \mathcal{T}\}$, say $\{\hat{D}_t : t \in \mathcal{T}\}$. Set $t = 1$.

Step 1.1 Solve the subproblem: for time period t solve (8) to get x_t.

Step 1.2 Apply the system dynamics: set

$$R^k_{j,t+1} = \sum_{i \in \mathcal{I}} \left(x^{ke}_{ijt} + x^{kl}_{ijt} \right) \qquad j \in \mathcal{I},\ k \in \mathcal{K}.$$

Step 1.3 Advance time: set $t = t + 1$. If $t \in \mathcal{T}$ go to step 1.1.

Step 2 Value function update: set $\hat{V}_t \leftarrow \mathcal{U}(\hat{V}_t, \hat{D}_t, R_t)$ for all $t \in \mathcal{T}$.

Step 3 Advance iteration counter: set $n = n + 1$. Go to step 1.

Figure 1 The Adaptive Dynamic Programming Algorithm

$(1-\alpha)R^2_t$, we have $V_t(R^0_t) \geq \alpha V_t(R^1_t) + (1-\alpha)V_t(R^2_t)$. Therefore, we propose using concave approximations of V_t, and for computational tractability we consider separable concave functions. For brevity, we refer to a separable, piecewise-linear, concave function as a *piecewise-linear* function for the rest of the paper.

In the presence of multiple vehicle types, (8) is a min-cost integer multicommodity-flow problem when we use piecewise-linear value-function approximations, and in this case, integer solutions may be hard to obtain. In this section, we come up with an approximation scheme using a special combination of linear and piecewise-linear functions that yields integer solutions naturally.

4.1. Linear Value-Function Approximations

First, we take our value-function approximation to be $\hat{V}_t(R_t) = \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} \hat{V}^k_{it}(R^k_{it})$, where each $\hat{V}^k_{it}$ is a linear function, say $\hat{V}^k_{it}(R^k_{it}) = \hat{v}^k_{it} R^k_{it}$. Then, (8) can be written as

$$\tilde{V}_t(R_t, D_t) = \max \sum_{i,j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \left(-c^{ke}_{ijt} x^{ke}_{ijt} + c^{kl}_{ijt} x^{kl}_{ijt} \right) + \sum_{j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \hat{v}^k_{j,t+1} R^k_{j,t+1} \tag{9}$$

$$\text{subject to} \quad (3), (4), (5)$$

$$\sum_{i \in \mathcal{I}} \left(x^{ke}_{ijt} + x^{kl}_{ijt} \right) - R^k_{j,t+1} = 0 \qquad j \in \mathcal{I},\ k \in \mathcal{K}. \tag{10}$$

By using (10), we can write (9) as

$$\tilde{V}(R_t, D_t) = \max \sum_{i,j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \left(\left(-c^{ke}_{ijt} + \hat{v}^k_{j,t+1} \right) x^{ke}_{ijt} + \left(c^{kl}_{ijt} + \hat{v}^k_{j,t+1} \right) x^{kl}_{ijt} \right)$$

$$\text{subject to} \quad (3), (4), (5),$$

which can be shown to be a min-cost network-flow problem.

4.2. Piecewise-Linear Value-Function Approximations

We now assume that $\hat{V}_t(R_t) = \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} \hat{V}^k_{it}(R^k_{it})$, where each $\hat{V}^k_{it}$ is a piecewise-linear, concave function, with the set of points of nondifferentiability being a subset of the set of positive integers.

Let $N^k = \sum_{i \in \mathcal{I}} R^k_{i1}$ be the total number of vehicles of type $k \in \mathcal{K}$ in the system. Clearly, the relevant domain of $\hat{V}^k_{it}$ is $\{0, 1, \ldots, N^k\}$. We can describe $\hat{V}^k_{it}$ by a sequence of numbers $\{\hat{v}^k_{it}(1), \hat{v}^k_{it}(2), \ldots, \hat{v}^k_{it}(N^k)\}$, where $\hat{v}^k_{it}(s)$ is the slope of $\hat{V}^k_{it}$ over $(s-1, s)$. The concavity of $\hat{V}^k_{it}$ dictates that $\hat{v}^k_{it}(1) \geq \hat{v}^k_{it}(2) \geq \cdots \geq \hat{v}^k_{it}(N^k)$. Then, (8) can be written as

$$\tilde{V}_t(R_t, D_t) = \max \sum_{i,j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \left(-c^{ke}_{ijt} x^{ke}_{ijt} + c^{kl}_{ijt} x^{kl}_{ijt} \right) + \sum_{j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{N}^k} \hat{v}^k_{j,t+1}(s) z^k_{j,t+1}(s) \tag{11}$$

$$\text{subject to} \quad (2), (3), (4), (5)$$

$$R^k_{j,t+1} - \sum_{s \in \mathcal{N}^k} z^k_{j,t+1}(s) = 0 \quad j \in \mathcal{I}, k \in \mathcal{K}$$

$$0 \leq z^k_{j,t+1}(s) \leq 1 \quad j \in \mathcal{I}, k \in \mathcal{K}, s \in \mathcal{N}^k, [-1pt]$$

where $\mathcal{N}^k = \{1, \ldots, N^k\}$. Problem (11) can be shown to be a min-cost integer multicommodity network-flow problem. However, its linear-programming relaxation turns out to be quite tight. In our experimental work, except for a few instances (which constitute fewer than 0.1% of the total instances we solve), the linear-programming relaxation of (11) yields integer solutions. Even if this is not the case, (11) is relatively small because it spans one time period, and the optimal solution can be found by exploring a few nodes in the branch-and-bound tree. Nevertheless, as shown in Section 5, the multicommodity nature of (11) brings an unwelcome dimension of complexity, especially when updating the value-function approximations. In the next section, we develop a new approximation scheme under which (8) is a min-cost network-flow problem.

4.3. Hybrid Value-Function Approximations

The interesting thing about piecewise-linear approximations is that they have a nice "damping" effect achieved by assigning decreasing marginal values to an incremental vehicle of a certain type in a certain location. Linear approximations lack this ability because they assign a constant marginal value to every incremental vehicle. The damping effect of piecewise-linear approximations prevents us from unnecessarily repositioning an excessive number of vehicles to a certain location. On the other hand, when working with linear approximations, we may unnecessarily reposition a high number of vehicles to a certain location if the slope of the value-function approximation corresponding to that location is large.

Hybrid value-function approximations try to reconstruct this damping effect while ensuring that the subproblems are min-cost network-flow problems. The core idea is as follows. Because the number of loaded movements is bounded by the number of demands available, we define a linear value-function approximation component for the vehicles that arrive at a location through a loaded movement, and a piecewise-linear value-function approximation component for the vehicles that arrive at a location through an empty movement. To formalize the idea, we set

$$X^{ke}_{j,t+1} = \sum_{i\in\mathcal{I}} x^{ke}_{ijt} \quad \text{and} \quad X^{kl}_{j,t+1} = \sum_{i\in\mathcal{I}} x^{kl}_{ijt},$$

with $X^e_{t+1} = [X^{ke}_{j,t+1}]_{j\in\mathcal{I},k\in\mathcal{K}}$ and $X^l_{t+1} = [X^{kl}_{j,t+1}]_{j\in\mathcal{I},k\in\mathcal{K}}$. Clearly, $R_{t+1} = X^e_{t+1} + X^l_{t+1}$ and V_{t+1} is a function of R_{t+1} rather than being a function of X^e_{t+1} and X^l_{t+1} separately. However, we adopt the form $\widehat{V}_{t+1}(R_{t+1}) = \widehat{W}_{t+1}(X^e_{t+1}) + \hat{L}_{t+1}(X^l_{t+1})$ for our approximations. The advantage of this approach is that if $\widehat{W}_{t+1}$ is piecewise-linear and $\hat{L}_{t+1}$ is linear, then (8) is a min-cost network-flow problem, and in Section 6, we show that this procedure provides better solution quality than do linear approximations, and is faster than piecewise-linear approximations.

Let us assume that

$$\widehat{W}_{t+1}(X^e_{t+1}) = \sum_{j\in\mathcal{I}}\sum_{k\in\mathcal{K}} \widehat{W}^k_{j,t+1}(X^{ke}_{j,t+1}),$$

where each $\widehat{W}^k_{j,t+1}$ is a piecewise-linear function defined by the sequence of slopes $\{\hat{w}^k_{j,t+1}(1), \hat{w}^k_{j,t+1}(2), \ldots, \hat{w}^k_{j,t+1}(N^k)\}$. Also, assume that $\hat{L}_{t+1}(X^l_{t+1}) = \sum_{j\in\mathcal{I}}\sum_{k\in\mathcal{K}} \hat{L}^k_{j,t+1}(X^{kl}_{j,t+1})$, where each $\hat{L}^k_{j,t+1}$ is a linear function with slope $\hat{l}^k_{j,t+1}$. Then (8) can be written as

$$\tilde{V}_t(R_t, D_t) = \max \sum_{i,j\in\mathcal{I}}\sum_{k\in\mathcal{K}} \left(-c^{ke}_{ijt}x^{ke}_{ijt} + c^{kl}_{ijt}x^{kl}_{ijt}\right) + \sum_{j\in\mathcal{I}}\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{N}^k} \hat{w}^k_{j,t+1}(s) z^k_{j,t+1}(s)$$

$$+ \sum_{j\in\mathcal{I}}\sum_{k\in\mathcal{K}} \hat{l}^k_{j,t+1} X^{kl}_{j,t+1} \tag{12}$$

subject to (3), (4), (5)

$$\sum_{i\in\mathcal{I}} x^{ke}_{ijt} - X^{ke}_{j,t+1} = 0 \quad j\in\mathcal{I},\ k\in\mathcal{K} \tag{13}$$

$$\sum_{i\in\mathcal{I}} x^{kl}_{ijt} - X^{kl}_{j,t+1} = 0 \quad j\in\mathcal{I},\ k\in\mathcal{K} \tag{14}$$

$$X^{ke}_{j,t+1} - \sum_{s\in\mathcal{N}^k} z^k_{j,t+1}(s) = 0 \quad j\in\mathcal{I},\ k\in\mathcal{K} \tag{15}$$

$$0 \le z^k_{j,t+1}(s) \le 1 \quad j\in\mathcal{I},\ k\in\mathcal{K},\ s\in\mathcal{N}^k. \tag{16}$$

Proposition 1. *Problem* (12) *is a min-cost network-flow problem.*

Proof. Use (14) to write $X^{kl}_{j,t+1}$ in the objective function as $\sum_{i\in\mathcal{I}} x^{kl}_{ijt}$. Combining the constraints (13) and (15), we obtain

$$\tilde{V}_t(R_t, D_t) = \max \sum_{i,j\in\mathcal{I}}\sum_{k\in\mathcal{K}} \left(-c^{ke}_{ijt}x^{ke}_{ijt} + \left(c^{kl}_{ijt} + \hat{l}^k_{j,t+1}\right)x^{kl}_{ijt}\right) + \sum_{j\in\mathcal{I}}\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{N}^k} \hat{w}^k_{j,t+1}(s) z^k_{j,t+1}(s) \tag{17}$$

subject to (3), (4), (5), (16)

$$\sum_{i\in\mathcal{I}} x^{ke}_{ijt} - \sum_{s\in\mathcal{N}^k} z^k_{j,t+1}(s) = 0 \quad j\in\mathcal{I},\ k\in\mathcal{K}. \tag{18}$$

Define variables $\{u_{ijt}\colon i, j\in\mathcal{I}\}$ and write (4) as

$$\sum_{k\in\mathcal{K}} x^{kl}_{ijt} - u_{ijt} = 0 \quad i,j\in\mathcal{I} \tag{19}$$

$$u_{ijt} \le D_{ijt} \quad i,j\in\mathcal{I}. \tag{20}$$

Then (17) becomes

$$\tilde{V}_t(R_t, D_t) = \max \sum_{i,j\in\mathcal{I}}\sum_{k\in\mathcal{K}} \left(-c^{ke}_{ijt}x^{ke}_{ijt} + \left(c^{kl}_{ijt} + \hat{l}^k_{j,t+1}\right)x^{kl}_{ijt}\right) + \sum_{j\in\mathcal{I}}\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{N}^k} \hat{w}^k_{j,t+1}(s) z^k_{j,t+1}(s) \tag{21}$$

subject to (3), (5), (16), (18), (19), (20).

Consider a network composed of three sets of nodes $\mathcal{H} = \{(i,k)\colon i\in\mathcal{I}, k\in\mathcal{K}\}$, $\mathcal{L} = \{(i,j)\colon i,j\in\mathcal{I}\}$, $\mathcal{M} = \{(j,k)\colon j\in\mathcal{I}, k\in\mathcal{K}\}$, and a root node. The supply of node $(i,k)\in\mathcal{H}$ is R^k_{it}. The supply of all the other nodes is zero except for the root node, which should have a supply of $-\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}} R^k_{it}$ to ensure feasibility. The variables in (21) correspond to the following arcs.

Variable	Tail node	Head node	Cost	Bound
$\{x^{ke}_{ijt}\colon i,j\in\mathcal{I}, k\in\mathcal{K}\}$	$(i,k)\in\mathcal{H}$	$(j,k)\in\mathcal{M}$	$-c^{ke}_{ijt}$	·
$\{x^{kl}_{ijt}\colon i,j\in\mathcal{I}, k\in\mathcal{K}\}$	$(i,k)\in\mathcal{H}$	$(i,j)\in\mathcal{L}$	$c^{kl}_{ijt} + \hat{l}^k_{j,t+1}$	·
$\{z^k_{j,t+1}(s)\colon j\in\mathcal{I}, k\in\mathcal{K}, s\in\mathcal{N}^k\}$	$(j,k)\in\mathcal{M}$	root node	$\hat{w}^k_{j,t+1}(s)$	1
$\{u_{ijt}\colon i,j\in\mathcal{I}\}$	$(i,j)\in\mathcal{L}$	root node	0	D_{ijt}

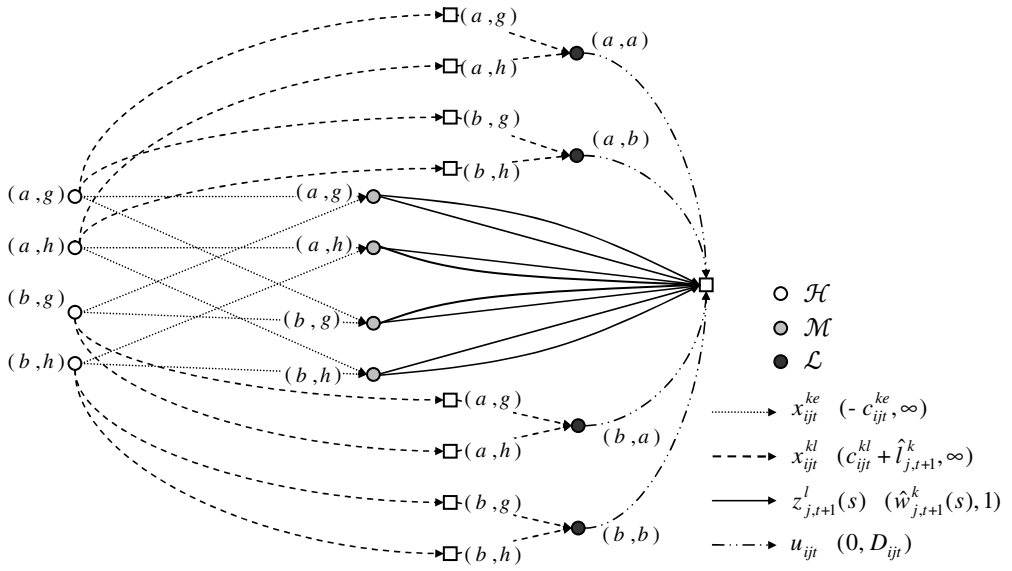

Figure 2 Network Representation of Problem (21)

Note. We assume that $\mathcal{I} = \{a, b\}$, $\mathcal{K} = \{g, h\}$. Node label $(i, k) \in \mathcal{I} \times \mathcal{K}$ denotes that node $(i, k) \in \mathcal{H}$ or $(i, k) \in \mathcal{M}$. Node label $(i, j) \in \mathcal{I}^2$ denotes that node $(i, j) \in \mathcal{L}$.

Constraints (3), (18), and (19) are respectively the flow-balance constraints for nodes $(i, k) \in \mathcal{H}$, $(j, k) \in \mathcal{M}$, and $(i, j) \in \mathcal{L}$. See Figure 2 for the graphical representation of the problem. □

5. Updating Value-Function Approximations

Let us assume that at iteration n, $\{\hat{D}_t^n\colon t \in \mathcal{T}\}$ is the sequence of demand realizations, $\{\hat{V}_t^n\colon t \in \mathcal{T}\}$ is the sequence of value-function approximations, and $\{R_t^n\colon t \in \mathcal{T}\}$ is the sequence of system states generated by solving approximate subproblems of the following form by using the current value-function approximations and demand realizations

$$\tilde{V}_t^n(R_t^n, \hat{D}_t^n) = \max_{(x_t, R_{t+1}) \in \mathcal{Y}(R_t^n, \hat{D}_t^n)} c_t x_t + \hat{V}_{t+1}^n(R_{t+1}). \quad (22)$$

Whether linear or piecewise-linear, we describe each value-function approximation component by a single slope (for linear) or a series of slopes (for piecewise-linear). Using e_i^k to denote the $|\mathcal{I}||\mathcal{K}|$ dimensional unit vector with one in the element corresponding to $i \in \mathcal{I}$, $k \in \mathcal{K}$, we would like to use $V_t(R_t^n + e_i^k, \hat{D}_t^n) - V_t(R_t^n, \hat{D}_t^n)$ and $V_t(R_t^n - e_i^k, \hat{D}_t^n) - V_t(R_t^n, \hat{D}_t^n)$ to update the slopes of the value-function approximation component $\hat{V}_{it}^{kn}$. However, this requires knowledge of the exact value-function. Instead, we propose using

$$\Phi_t^n(\mp e_i^k) = \tilde{V}_t^n(R_t^n \mp e_i^k, \hat{D}_t^n) - \tilde{V}_t^n(R_t^n, \hat{D}_t^n). \quad (23)$$

Here, we see another advantage of having min-cost network-flow subproblems. For subproblems with min-cost integer multicommodity-flow structure, computing $\{\Phi_t^n(\mp e_i^k)\colon i \in \mathcal{I}, k \in \mathcal{K}\}$ requires solving $2|\mathcal{I}||\mathcal{K}| + 1$ subproblems of form (22). However, if the subproblems are min-cost network-flow problems, then $\{\Phi_t^n(\mp e_i^k)\colon i \in \mathcal{I}, k \in \mathcal{K}\}$ can be computed by using flow-augmenting-decreasing paths, and this amounts to executing a shortest-path algorithm only twice (Powell 1989).

5.1. Updating Linear Value-Function Approximations

We assume that each linear value-function approximation component $\hat{V}_{it}^{kn}$ is characterized by the slope $\hat{v}_{it}^{kn}$. Then, we set

$$\hat{v}_{it}^{k,n+1} = (1 - \alpha^n)\hat{v}_{it}^{kn} + \alpha^n \Phi_t^n(e_i^k) \quad (24)$$

to obtain the slope of the value-function approximation component $\hat{V}_{it}^{k,n+1}$, where $\alpha^n \in (0, 1)$ is the step size at iteration n. We favor using $\Phi_t^n(e_i^k)$ over $\Phi_t^n(-e_i^k)$ merely because $\Phi_t^n(-e_i^k)$ is not defined when $R_{it}^{kn} = 0$.

5.2. Updating Piecewise-Linear Value-Function Approximations

We now assume that each piecewise-linear value-function approximation component $\hat{V}_{it}^{kn}$ is characterized by a sequence of slopes $\{\hat{v}_{it}^{kn}(1), \hat{v}_{it}^{kn}(2), \ldots, \hat{v}_{it}^{kn}(N^k)\}$. We update the value-function approximation component $\hat{V}_{it}^{kn}$ as follows.

1. Set

$$q_{it}^{kn}(s) = \begin{cases} (1 - \alpha^n)\hat{v}_{it}^{kn}(s) + \alpha^n(-\Phi_t^n(-e_i^k)) & \text{for } s = R_{it}^{kn} \\ (1 - \alpha^n)\hat{v}_{it}^{kn}(s) + \alpha^n \Phi_t^n(e_i^k) & \text{for } s = R_{it}^{kn} + 1 \\ \hat{v}_{it}^{kn}(s) & \text{for } s \in \mathcal{N}^k \backslash \{R_{it}^{kn}, R_{it}^{kn} + 1\}. \end{cases}$$

2. Set

$$\hat{v}_{it}^{k,n+1} = \arg\min_{z} \left\{ \sum_{s \in \mathcal{N}^k} \left(z(s) - q_{it}^{kn}(s)\right)^2 : z(s) \geq z(s+1) \quad s = 1, \ldots, N^k - 1 \right\},$$

where $\hat{v}_{it}^{k,n+1}$ and z refer to the vectors $[\hat{v}_{it}^{k,n+1}(s)]_{s \in \mathcal{N}^k}$ and $[z(s)]_{s \in \mathcal{N}^k}$ respectively.

The first step is very similar to (24), except for the fact that the update applies only to the "relevant" domain of the function. However, after this step, a piecewise-linear function described by the sequence of slopes $\{q_{it}^{kn}(1), q_{it}^{kn}(2), \ldots, q_{it}^{kn}(N^k)\}$ is not necessarily concave. Hence in the second step, we project the function defined by the sequence of slopes $\{q_{it}^{kn}(1), q_{it}^{kn}(2), \ldots, q_{it}^{kn}(N^k)\}$ onto the set of piecewise-linear concave functions with sets of points of nondifferentiability being a subset of positive integers. The procedure above is due to Powell et al. (2002), who show that making this projection involves a very simple computation and prove converge properties for two-stage problems with separable value functions.

5.3. Implications of Multiperiod Travel Times

As we pointed out in Section 2, extending our formulation to the case where not all movements take a single time period is straightforward. Under multiperiod travel times, the state vector is

$$R_t = [R_{it't}^k]_{i \in \mathcal{I},\, k \in \mathcal{K},\, t \leq t' < t + \tau_{\max}},$$

where $\tau_{\max}$ is the maximum travel time possible.

Therefore, we can define our separable value-function approximations as

$$V_t(R_t) = \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} \sum_{t'=t}^{t+\tau_{\max}-1} \hat{V}_{it't}^k(R_{it't}^k).$$

In Figure 3, we show the performance of our solution approach on two problems with deterministic demand arrival processes. Because the demand arrivals are deterministic, these problems can be formulated as in (1) (with trivial modifications in order to accommodate multiperiod travel times). In the chart, 100% refers to the objective value of the linear-programming relaxation of this problem, and hence, is an upper bound on the optimal objective value. Comparison of the data series labeled "single-period travel times" and "multiperiod travel times, regular performance" shows that it may take many more iterations for our approach to provide a good solution for problems with multiperiod travel times. In this section, we derive a result that enhances the performance of our approach in the presence of multiperiod travel times. This result will increase the performance of our approach to the data series labeled "multiperiod travel times, accelerated performance."

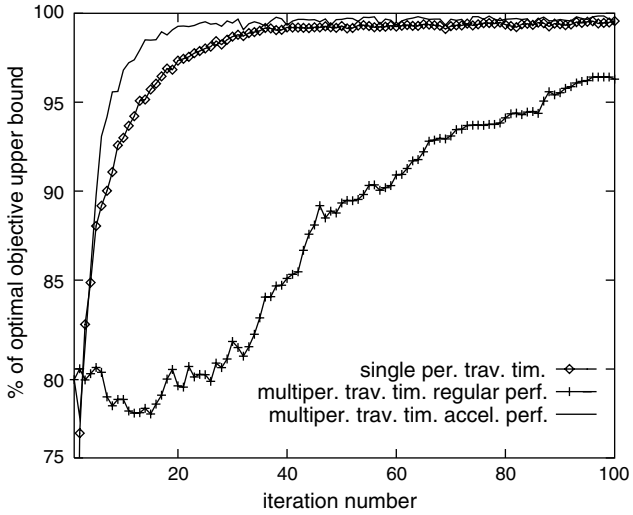

Figure 3 **Comparison of the Performance of the Algorithm on Problems with Single-Period and Multiperiod Travel Times**

For a given resource state vector R_t and demand realizations $\{\hat{D}_t, \ldots, \hat{D}_T\}$, we define problem $PR_t(R_t, \hat{D}_t, \ldots, \hat{D}_T)$ and its optimal objective value $F_t(R_t, \hat{D}_t, \ldots, \hat{D}_T)$ as

$$\begin{aligned}
&F_t(R_t, \hat{D}_t, \ldots, \hat{D}_T) \\
&\quad = \max \sum_{s=t}^{T} \sum_{i,\, j \in \mathcal{I}} \sum_{k \in \mathcal{K}} \left(-c_{ijs}^{ke} x_{ijs}^{ke} + c_{ijs}^{kl} x_{ijs}^{kl}\right) \\
&\text{subject to } \sum_{j \in \mathcal{I}} \left(x_{ijs}^{ke} + x_{ijs}^{kl}\right) - R_{iss}^k = 0 \\
&\qquad\qquad i \in \mathcal{I},\ k \in \mathcal{K},\ t \leq s \leq T \\
&\qquad \sum_{i \in \mathcal{I}} \left(\mathbf{1}_{\tau_{ij}}(t'-s) x_{ijs}^{ke} + \mathbf{1}_{\tau_{ij}}(t'-s) x_{ijs}^{kl}\right) \\
&\qquad\qquad + R_{jt's}^k - R_{jt',s+1} = 0 \\
&\qquad\qquad j \in \mathcal{I},\ k \in \mathcal{K},\ t \leq s \leq T, \\
&\qquad\qquad s+1 \leq t' \leq s + \tau_{\max} \\
&\qquad \sum_{k \in \mathcal{K}} x_{ijs}^{kl} \leq \hat{D}_{ijs} \quad i, j \in \mathcal{I},\ t \leq s \leq T \\
&\qquad\qquad x_{ijs}^{ke},\ x_{ijs}^{kl} \in \mathbb{Z}_+ \quad i, j \in \mathcal{I},\ k \in \mathcal{K},\ t \leq s \leq T.
\end{aligned}$$

The problem above can be viewed as the *posterior multistage* problem corresponding to demand realization $\{\hat{D}_t, \ldots, \hat{D}_T\}$. Let $\{R_t^*: t \in \mathcal{T}\}$ and $\{x_t^*: t \in \mathcal{T}\}$ be the optimal solution to $PR_1(R_1, \hat{D}_1, \ldots, \hat{D}_T)$.

For the remainder of this section we denote the vector obtained by increasing the element corresponding to $i \in \mathcal{I}$, $k \in \mathcal{K}$ and t' of R_t^* by one as $R_t^* + e_{it'}^k$. Therefore $e_{it'}^k$ can be visualized as a unit vector of dimension $|\mathcal{I}||\mathcal{K}|\tau_{\max}$. However, the exact place of one becomes clear when it is used together with R_t^*. We have the following result.

Proposition 2. *For all $i \in \mathcal{I}$, $k \in \mathcal{K}$, $1 \le t \le T-1$, and $t < t' < t + \tau_{\max}$, we have*

$$\begin{aligned} &F_t(R_t^* \mp e_{it'}^k, \hat{D}_t, \ldots, \hat{D}_T) - F_t(R_t^*, \hat{D}_t, \ldots, \hat{D}_T) \\ &\quad \ge F_{t+1}(R_{t+1}^* \mp e_{it'}^k, \hat{D}_{t+1}, \ldots, \hat{D}_T) \\ &\qquad - F_{t+1}(R_{t+1}^*, \hat{D}_{t+1}, \ldots, \hat{D}_T). \end{aligned}$$

Proof. We show only the case when the coefficient of $e_{it'}^k$ is positive.

We need to make three observations.

Observation 1. Because $\{R_t^*: t \in \mathcal{T}\}$ and $\{x_t^*: t \in \mathcal{T}\}$ are the optimal solution to $PR_1(R_1, \hat{D}_1, \ldots, \hat{D}_T)$, we have

$$\begin{aligned} &F_t(R_t^*, \hat{D}_t, \ldots, \hat{D}_T) \\ &\quad = c_t x_t^* + F_{t+1}(R_{t+1}^*, \hat{D}_{t+1}, \ldots, \hat{D}_T). \end{aligned} \tag{25}$$

Observation 2. From the first observation, x_t^* is a part of a feasible (and optimal) solution to $PR_t(R_t^*, \hat{D}_t, \ldots, \hat{D}_T)$. Therefore x_t^* is a part of a feasible (but not necessarily optimal) solution to $PR_t(R_t^* + e_{it'}^k, \hat{D}_t, \ldots, \hat{D}_T)$.

Observation 3. Applying decisions x_t^* on R_t^* generates the state vector R_{t+1}^*. Thus, applying decisions x_t^* on $R_t^* + e_{it'}^k$ generates the state vector $R_{t+1}^* + e_{it'}^k$. By observations 2 and 3, we have

$$\begin{aligned} &F_t(R_t^* + e_{it'}^k, \hat{D}_t, \ldots, \hat{D}_T) \\ &\quad \ge c_t x_t^* + F_{t+1}(R_{t+1}^* + e_{it'}^k, \hat{D}_{t+1}, \ldots, \hat{D}_T). \end{aligned}$$

The result is obtained by subtracting (25) from the inequality above. □

By repeated application of Proposition 2, we get

$$\begin{aligned} &F_t(R_t^* + e_{it'}^k, \hat{D}_t, \ldots, \hat{D}_T) - F_t(R_t^*, \hat{D}_t, \ldots, \hat{D}_T) \\ &\quad \ge F_{t+1}(R_{t+1}^* + e_{it'}^k, \hat{D}_{t+1}, \ldots, \hat{D}_T) \\ &\qquad - F_{t+1}(R_{t+1}^*, \hat{D}_{t+1}, \ldots, \hat{D}_T) \\ &\quad \ge \cdots \ge F_{t'-1}(R_{t'-1}^* + e_{it'}^k, \hat{D}_{t'-1}, \ldots, \hat{D}_T) \\ &\qquad\qquad - F_{t'-1}(R_{t'-1}^*, \hat{D}_{t'-1}, \ldots, \hat{D}_T) \\ &\quad \ge F_{t'}(R_{t'}^* + e_{it'}^k, \hat{D}_{t'}, \ldots, \hat{D}_T) \\ &\qquad - F_{t'}(R_{t'}^*, \hat{D}_{t'}, \ldots, \hat{D}_T). \end{aligned} \tag{26}$$

In effect, (26) states that the marginal value of a vehicle of type $k \in \mathcal{K}$ inbound to location $i \in \mathcal{I}$ and time period t' decreases over time. A similar chain of inequalities can be written for $F_t(R_t^* - e_{it'}^k, \hat{D}_t, \ldots, \hat{D}_T) - F_t(R_t^*, \hat{D}_t, \ldots, \hat{D}_T)$, which proves the following result.

Proposition 3. *For all $i \in \mathcal{I}$, $k \in \mathcal{K}$, $t \in \mathcal{T}$, and $t \le t' < t + \tau_{\max}$, we have*

$$\begin{aligned} &F_t(R_t^* \mp e_{it'}^k, \hat{D}_t, \ldots, \hat{D}_T) - F_t(R_t^*, \hat{D}_t, \ldots, \hat{D}_T) \\ &\quad = \max_{s \in \{t, t+1, \ldots, t'\}} \{F_s(R_s^* \mp e_{it'}^k, \hat{D}_s, \ldots, \hat{D}_T) \\ &\qquad - F_s(R_s^*, \hat{D}_s, \ldots, \hat{D}_T)\}. \end{aligned}$$

At the beginning of Section 5, we proposed using $\tilde{V}_t^n(R_t^n \mp e_{it'}^k, \hat{D}_t^n) - \tilde{V}_t^n(R_t^n, \hat{D}_t^n)$ to update the value-function approximation component $\hat{V}_{it't}^k$ at iteration n. This updating scheme yields the slow performance mentioned at the beginning of this section. Motivated by the previous proposition, we propose using

$$\max_{s \in \{t, t+1, \ldots, t'\}} \{\tilde{V}_s^n(R_s^n \mp e_{it'}^k, \hat{D}_s^n) - \tilde{V}_s^n(R_s^n, \hat{D}_s^n)\}.$$

Note that to update the value-function approximation for time period t, the new procedure uses information coming from time periods $t, t+1, \ldots, t+\tau_{\max}-1$. The data series labeled "multiperiod travel times, accelerated performance" in Figure 3 show that the performance of the algorithm on problems with multiperiod travel times becomes comparable to the performance of the algorithm on problems with single-period travel times when we use this new updating procedure.

6. Experimental Results

In this section, we explore the effectiveness of three different value-function approximation strategies on the jet-management problem described in Section 2. For our numerical work, we drop the two assumptions adopted throughout the paper and assume that there are multiple customer types and multiperiod travel times.

We present two sets of experiments. The first one includes problems with deterministic demand-arrival processes, which can also be formulated as large integer programs to obtain a tight bound on their optimal objective values. The second set of experiments is on problems with stochastic demand-arrival processes and compare our approach with a common engineering strategy that uses point forecasts of future demand realizations.

6.1. Experimental Setup

Our experimental strategy is to create one base problem and modify its different attributes to come up with problems with different characteristics.

We take $c_{ijt}^{ke} = c\delta_{ij}$, where c is the empty repositioning cost per mile and δ_{ij} is the distance from location i to j, and $c_{ijt}^{kdl} = rC_{kd}\delta_{ij}$, where r is the profit per loaded mile, and $C_{kd} \in [0, 1]$ denotes the suitability of a vehicle of type k to cover a demand from customer d. Therefore, for every customer type there are more (and less) preferred vehicle types. We generate the number of demands on each origin-destination pair and at every time period from the Poisson distribution with different means. We pay attention to generate date sets such that the number of demands outbound from a

Table 1 Characteristics of the Test Problems Used in the Experiments

Problem	T	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{D}\rvert$	R	D	c	r	C
Base	60	20	5	5	200	4,000	4	5	I
T_30	30	20	5	5	200	2,000	4	5	I
T_90	90	20	5	5	200	6,000	4	5	I
I_10	60	10	5	5	200	4,000	4	5	I
I_40	60	40	5	5	200	4,000	4	5	I
C_II	60	20	5	5	200	4,000	4	5	II
C_III	60	20	5	5	200	4,000	4	5	III
C_IV	60	20	5	5	200	4,000	4	5	IV
R_1	60	20	5	5	1	4,000	4	5	I
R_5	60	20	5	5	5	4,000	4	5	I
R_400	60	20	5	5	400	4,000	4	5	I
R_800	60	20	5	5	200	4,000	4	5	I
c_1.6	60	20	5	5	200	4,000	1.6	5	I
c_8	60	20	5	5	200	4,000	8	5	I

Notes. $R = \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}} R_{i1}^k$, $D = \mathbb{E}[\sum_{t\in\mathcal{T}} \sum_{i,\,j\in\mathcal{I}} \sum_{d\in\mathcal{D}} D_{ijt}^d]$. See Table 2 for different values of C.

location is negatively correlated with the number of demands inbound to a location. We expect these problems to require plenty of empty repositioning moves in the optimal solution, and hence, to be harder to solve. Table 1 summarizes the attributes of the test problems used in our experimental work. Different values used for $C = [C_{kd}]_{k\in\mathcal{K},\,d\in\mathcal{D}}$ are given in Table 2.

6.2. Deterministic Experiments

In our deterministic experiments, we formulate each problem as a min-cost integer multicommodity-flow problem similar to (1) and solve its linear-programming relaxation. This provides an upper bound on the optimal value of the objective function. We run the three different value-function approximation strategies described in Section 4 for 150 iterations using the step size $\alpha^n = 20/(40 + n)$ at iteration n. We concentrate on the objective-function value of the last 50 iterations to eliminate the effect of the "warming up" period of the value-function approximations.

The results are shown in Table 3. L, P, and PL refer to linear, piecewise-linear, and hybrid value-function approximation strategies described in Sections 4.1, 4.2, and 4.3, respectively. In the table, we give the average and maximum objective value in the last 50 iterations, and the number of iterations required to reach a certain percentage of the upper bound on the optimal objective value. The objective values we present are normalized to 100 by using the objective value of the linear-programming relaxation. The following observations can be made.

(1) P and PL yield results within 1%–2% of the upper bound. Considering its CPU-time advantage, PL seems to be an excellent approach for deterministic problems.

(2) PL takes more iterations to reach a given solution quality, but it makes up for this difference by faster runtime per iteration. Despite the fact that both L and PL solve min-cost network-flow subproblems, PL takes considerably more time per iteration due to the additional computational burden brought by updating the piecewise-linear approximations and setting up the subproblems.

(3) Figure 4 shows the typical performance of the three approximations. The objective value provided by L fluctuates from one iteration to the next, whereas P and PL yield more stable behavior.

(4) Although the presence of multiple types of vehicles that can exist in the same location and compete to serve the same demands destroys the separability of the value functions to a great extent, P and PL both yield high-quality solutions for all different substitution patterns.

(5) The substitution pattern affects the CPU time per iteration for P. This suggests that the tightness of the linear-programming relaxations of the min-cost integer multicommodity-flow subproblems solved by P depends on the substitution pattern.

(6) If the number of available vehicles is very small compared to $|\mathcal{I}||\mathcal{K}|$, on the average, we would expect zero or one vehicles of each type in each location. In this case, only the "initial slopes" of the piecewise-linear approximations would be utilized and L should be as effective as P. Problems R_1 and R_5 (where there is approximately one vehicle of each type) verify this expectation.

(7) Even if no empty repositioning is allowed, the exact value function is nonseparable due to interactions in the future time periods and vehicles moving from one location to another by serving demands. However, in general we expect separable approximations to work better on problems with high repositioning costs because they are less likely to require empty repositioning. P and PL perform well under all different repositioning costs, and low repositioning cost is not a problem.

Table 2 Different Substitution Patterns Used in the Experiments

I					II					III					IV				
1	0.8	0.5	0.3	0	1	0	0	0	0	1	0.5	0	0	0	1	1	1	1	1
0.7	1	0.8	0.3	0	1	1	0	0	0	0.5	1	0.5	0	0	1	1	1	1	1
0.6	0.6	1	0.5	0.5	1	1	1	0	0	0	0.5	1	0.5	0	1	1	1	1	1
0	0.4	0.7	1	0.5	1	1	1	1	0	0	0	0.5	1	0.5	1	1	1	1	1
0	0.4	0.6	0.6	1	1	1	1	1	1	0	0	0	0.5	1	1	1	1	1	1

Table 3 Results of Deterministic Runs

	Max. obj. value			Mean obj. value			Number of iterations to reach									Time (s) per iteration		
							L			P			PL					
Prob.	L	P	PL	L	P	PL	90	95	97.5	90	95	97.5	90	95	97.5	L	P	PL
Base	90.4	98.3	99.5	89.9	98.1	99.3	52			4	8	13	6	13	22	8	49	13
T_30	91.7	98.4	99.7	89.2	98.3	99.3	40			3	12	17	5	11	22	5	26	6
T_90	90.5	98.6	99.3	88.3	98.5	99.0	44			3	7	14	5	11	16	13	71	19
I_10	93.8	99.7	99.8	92.3	99.5	99.7	15			2	4	8	4	5	11	2	12	6
I_40	86.6	99.0	99.0	85.2	98.7	98.9				7	10	19	9	16	34	22	141	40
C_II	91.3	99.4	98.8	90.1	99.2	98.5	41			2	6	12	5	11	22	7	82	15
C_III	88.5	98.7	99.7	87.2	98.6	99.7				3	13	19	5	13	24	6	39	9
C_IV	89.3	99.7	98.9	84.9	99.7	98.7				4	7	12	4	14	26	7	75	18
R_1	99.4	98.3	99.3	96.1	94.9	96.2	3	4	4	7	11	23	3	4	4	5	22	8
R_5	97.1	97.4	95.0	95.2	94.2	93.3	10	32		3	9		42	61		6	48	9
R_100	91.6	97.7	97.2	90.1	97.6	97.0	50			14	16		10	44		7	47	9
R_400	91.6	99.3	99.5	88.5	99.1	99.5	52			4	6	14	4	8	20	8	49	14
c_1.6	88.1	99.6	99.9	85.9	99.3	99.9				2	3	17	3	3	9	8	47	12
c_8	92.9	98.3	98.6	91.1	98.2	98.5	45			8	14	24	10	21	39	8	45	12

6.3. Stochastic Experiments

Testing our solution approach for stochastic problems involves two stages. The first stage is called the *training iterations* where at each iteration we sample one demand realization, solve approximate subproblems of form (22), and update the value-function approximations. The second stage is called the *testing iterations* where we fix the value-function approximations and test the solution quality provided by the current set of approximations for different demand realizations. In our experiments we use 100 training and 250 testing iterations.

The benchmark we use is the common engineering practice referred to as the *rolling-horizon* (RH) procedure and it uses deterministic forecasts of the future demand realizations. An n-period RH solves an n-period deterministic integer program for every time period $t \in \mathcal{T}$. The first time period of this problem uses the demand realizations of the current scenario at time period t and the next $n-1$ time periods use the expected values of the demand random variables for time periods $t+1, \ldots, t+n-1$. Having solved this large integer program, we implement the decisions of the first time period (time period corresponding to t). We then proceed to solve the problem for time period $t+1$. To choose a value for n, we apply RH on problem Base for different values of n. As Table 4 shows, the performance of RH stabilizes when n is around 12 to 20. We use $n = 20$.

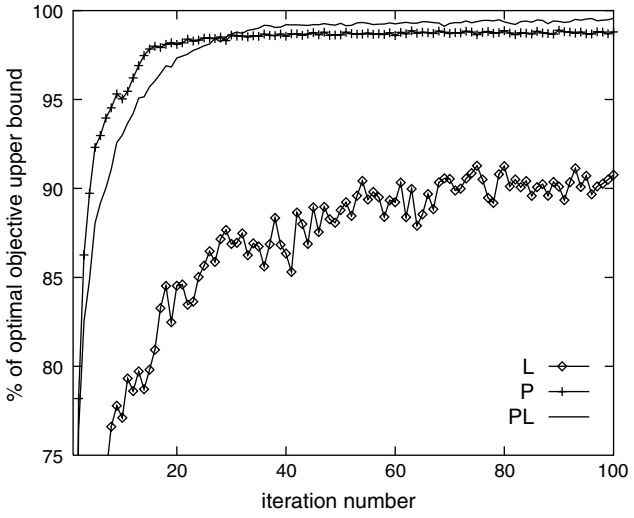

Figure 4 Typical Performances of L, P, and PL on a Deterministic Problem

Table 5 summarizes the results of the stochastic experiments, where in the first two sets of columns, we give the mean and the standard deviation of the objective value in 250 testing iterations. Furthermore, for every testing iteration we give a rank to each solution method depending on the objective value it yields for that particular testing iteration (one being the best-performing solution method). In the third set of columns, we give the average ranking of each method. We scale the objective value of each iteration by dividing it by a constant so that the objective values are on the order of 100. We make the following observations.

(1) PL performs significantly worse than does P in all problem instances. This behavior, contradicting

Table 4 Performance of RH on Problem Base with Different Rolling Horizon Lengths

n	4	8	12	16	20	24
Mean	83.4	87.3	88.8	88.9	88.9	88.8

Table 5 Results of Stochastic Runs

	Mean objective value				Standard deviation				Mean rank			
Problem	L	P	PL	RH	L	P	PL	RH	L	P	PL	RH
Base	80.5	95.5	90.8	88.9	2.5	2.2	2.0	1.9	4.0	1.0	2.0	3.0
I_10	84.5	95.2	93.5	91.5	2.3	2.4	2.4	2.4	4.0	1.0	2.0	3.0
I_40	74.1	92.2	87.3	86.9	1.8	0.5	0.6	0.8	4.0	1.0	2.0	3.0
C_II	80.6	95.4	89.9	90.8	2.1	2.1	2.1	2.0	4.0	1.0	2.9	2.1
C_III	77.7	95.6	89.5	86.3	2.1	2.2	2.1	1.9	4.0	1.0	2.0	3.0
C_IV	82.8	95.5	92.4	92.2	2.6	2.2	2.1	2.2	4.0	1.0	2.3	2.7
R_1	88.9	85.2	73.2	52.9	7.6	8.7	9.0	8.4	1.0	2.0	3.0	4.0
R_5	91.8	86.6	65.8	54.3	5.0	5.2	5.7	6.0	1.0	2.0	3.0	4.0
R_100	83.7	95.8	82.4	86.7	2.1	2.1	1.9	2.0	3.1	1.0	3.9	2.0
R_400	83.0	95.0	93.9	90.2	2.2	2.2	2.1	2.0	4.0	1.0	2.0	3.0
c_1.6	73.8	95.6	94.4	90.6	2.1	2.1	2.0	1.9	4.0	1.0	2.0	3.0
c_8	84.7	94.9	92.7	88.7	2.7	2.3	2.3	2.2	4.0	1.0	2.0	3.0

our findings in the deterministic experiments, can be explained as follows. We are trying to approximate the nonseparable value function with a separable function. To get a good solution to a deterministic problem instance, it is enough to capture the shape of the value function in the vicinity of the optimal solution. However, for problems with stochastic demand arrivals, we need to capture the shape of the value function over a wider range (what we mean by "range" is a subset of the domain of the value function), and the linear components of PL prevent us from doing this effectively.

(2) P provides better results than does RH on all problem instances, and PL provides solutions that are slightly better than those of RH on a majority of the problems.

(3) The CPU times for different approximation strategies are comparable to those for deterministic experiments. The only computational burden brought by stochastic load-arrival processes is the need to draw a new sample realization at the beginning of each iteration.

(4) In general, linear approximations yield poor solution quality. However, their fast runtimes may make them attractive in the early iterations to initialize the piecewise-linear approximations, and then we can switch to piecewise-linear or hybrid approximation strategies.

References

Aneja, Y. P., K. P. Nair. 1982. Multicommodity network flows with probabilistic loses. *Management Sci.* **28** 1080–1086.

Assad, A. A. 1978. Multicommodity network flows—A survey. *Networks* **8** 37–91.

Bellman, R. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

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

Chen, Z.-L., W. B. Powell. 1999. A convergent cutting-plane and partial-sampling algorithm for multistage linear programs with recourse. *J. Optim. Theory Appl.* **103** 497–524.

Chih, K.-K. 1986. A real time dynamic optimal freight car management simulation model of the multiple railroad, multicommodity temporal spatial network flow problem. Ph.D. thesis, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ.

Crainic, T. G., M. Gendreau, P. Dejax. 1993. Dynamic and stochastic models for the allocation of empty containers. *Oper. Res.* **41** 102–126.

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

Hane, C., C. Barnhart, E. Johnson, R. Marsten, G. Nemhauser, G. Sigismondi. 1995. The fleet assignment problem: Solving a large-scale integer program. *Math. Programming* **70** 211–232.

Haneveld, W. K. K., M. H. van der Vlerk. 1999. Stochastic integer programming: General models and algorithms. *Ann. Oper. Res.* **85** 39–57.

Higle, J., S. Sen. 1991. Stochastic decomposition: An algorithm for two stage linear programs with recourse. *Math. Oper. Res.* **16** 650–669.

Holmberg, K., M. Joborn, J. T. Lundgren. 1998. Improved empty freight car distribution. *Transportation Sci.* **32** 163–173.

Jordan, W., M. Turnquist. 1983. A stochastic dynamic network model for railroad car distribution. *Transportation Sci.* **17** 123–145.

Kennington, J. L. 1978. A survey of linear cost multicommodity network flows. *Oper. Res.* **26** 209–236.

Powell, W. B. 1989. A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy. *Transportation Sci.* **23** 231–243.

Powell, W. B., T. A. Carvalho. 1998. Dynamic control of logistics queueing network for large-scale fleet management. *Transportation Sci.* **32** 90–109.

Powell, W. B., P. Jaillet, A. Odoni. 1995. Stochastic and dynamic networks and routing. M. O. Ball, T. L. Magnanti, C. L. Monma, G. L. Nemhauser, eds. *Network Routing, Handbooks in Operations Research and Management Science*, Vol. 8. North-Holland, Amsterdam, The Netherlands, 141–295.

Powell, W. B., A. Ruszczynski, H. Topaloglu. 2004. Learning algorithms for separable approximations of discrete stochastic optimization problems. *Math. Oper. Res.* **29** 814–836.

Shan, Y. 1985. A dynamic multicommodity network flow model for real-time optimal rail freight car management. Ph.D. thesis, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ.

Soroush, H. P., B. Mirchandani. 1990. The stochastic multicommodity flow problem. *Networks* **20** 121–155.