An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management, II: Multiperiod Travel Times

Gregory A. Godfrey • Warren B. Powell
Department of Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey 08544

In a companion paper (Godfrey and Powell 2002) we introduced an adaptive dynamic programming algorithm for stochastic dynamic resource allocation problems, which arise in the context of logistics and distribution, fleet management, and other allocation problems. The method depends on estimating separable nonlinear approximations of value functions, using a dynamic programming framework. That paper considered only the case in which the time to complete an action was always a single time period. Experiments with this technique quickly showed that when the basic algorithm was applied to problems with multiperiod travel times, the results were very poor. In this paper, we illustrate why this behavior arose, and propose a modified algorithm that addresses the issue. Experimental work demonstrates that the modified algorithm works on problems with multiperiod travel times, with results that are almost as good as the original algorithm applied to single period travel times.

In Godfrey and Powell (2002), we introduced an adaptive dynamic programming algorithm for stochastic dynamic resource allocation problems. These problems arise in the context of inventory distribution problems, fleet management (such as rail, trucking, and container transportation), and personnel management. The algorithm used nonlinear approximations of the value of resources in the future. It produced near-optimal solutions when applied to deterministic problems, and solutions that significantly outperformed rolling-horizon procedures when applied to stochastic versions of the problem. The nonlinear value functions were estimated using the CAVE algorithm of Godfrey and Powell (2001), which uses stochastic samples of left and right gradients to build a piecewise-linear concave approximation of the value function.

In this first application of the CAVE algorithm to multistage problems, we assumed that any decision acting on a resource would be completed in the next time period. In transportation applications this is equivalent to assuming that the travel time between any pair of cities is one time period. Such an assumption might be quite realistic in the context of a distribution problem where the time period is one week, but it is quite limiting in settings where the time periods are much smaller.

The issue that arises with multiperiod travel times is unique to nonlinear approximations of the value function. In earlier work, Powell and Carvalho (1998) used linear approximations that work equally well for both single and multiperiod travel times. The nonlinear approximation, as shown in Godfrey and Powell (2002), outperforms the linear approximation; the algorithm is much faster and more stable, and produces solutions with noticeably better quality. When the same algorithm was applied to problems with multiperiod travel times, however, the initial results were quite poor. In fact, they were much worse than the linear approximation.

The problem is illustrated in Figure 1, where we show three points in space and time. We are

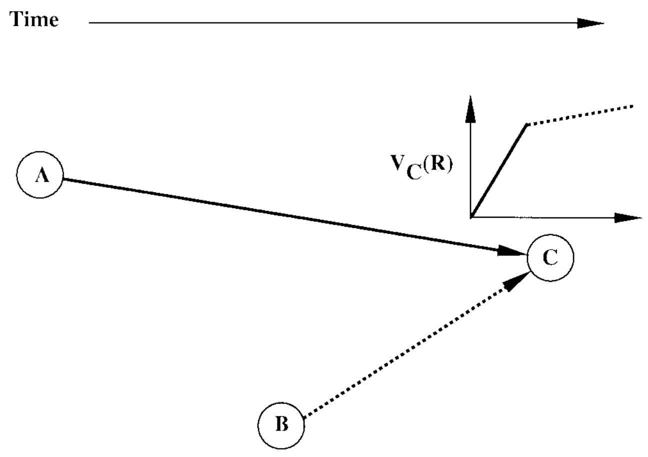

Figure 1 **Illustration of the Error Introduced Using Nonlinear Functions and Multiperiod Travel Times**

considering sending vehicles from space-time points A and B, both of which will arrive at space-time point C. Since A is farther away, we make this decision first. When we make the decision, we consider the cost of moving from A to C plus the value of the resources at C, represented by our value function approximation $V_C(R)$, where R is the total flow of resources into point C (measured when we make the decision). Because A is farthest away, location A "sees" the steepest slope on the function (shown as a solid line, just like the line from A to C). When we get to location B, we have already made the decision to send flow from A to C, so location B "sees" the shallower slope, indicated by the dotted line (again, drawn similarly to the line from B to C). In a fleet management problem, long travel times also mean larger costs. If A had excess vehicles and C appeared to be short on vehicles, it seems reasonable for location A to reposition vehicles empty from A to C, paying the higher cost in return for the apparently higher reward that would be received at C.

We refer to this behavior as the "long-haul bias" but claim that it is a surprisingly common phenomenon. For example, the FAA in the United States will be asked to clear a flight for takeoff at noon from Los Angeles to New York, arriving at 6 pm. The air-traffic controller might look at the total number of aircraft expected to arrive at 6 pm when the request is made, and conclude that the airport looks pretty clear at that time. Five hours later, a flight from Boston requests permission to take off, also expecting to arrive in New York at 6 pm. By this time, the controller has already given permission to enough aircraft that the anticipated number of arrivals to New York exceeds the capacity of the airport.

We face these types of decisions in our own scheduling. Consider, for example, when someone is asked to give a talk at a conference two years into the future. The conference may not be a major one, but your schedule is undoubtedly clear at that time. You agree to give the talk, but as the conference approaches, your schedule starts to fill in with more important activities, and you begin to wish you had not made the commitment.

This paper makes the following contributions. First we identify, apparently for the first time, the problem with using nonlinear value functions in the context of stochastic resource-allocation problems where decisions require more than one time period to complete. Second, we propose a solution strategy and a computationally tractable algorithm that addresses this problem. Third, we show experimentally that the algorithm provides near-optimal solutions when applied to deterministic problems, and solutions that significantly outperform deterministic rolling-horizon procedures for stochastic problems.

The paper is organized as follows. Section 1 presents the notation and formulates the stochastic optimization problem. Section 2 describes a solution strategy that is based on formulating the problem as a dynamic program using the concept of an incomplete state variable. This section describes in greater detail the challenge of using nonlinear functional approximations in the context of multiperiod travel times, and proposes a new method that handles the multiperiod issue. Finally, §3 presents a series of experiments that demonstrates the effectiveness of the new technique on both deterministic and stochastic problems.

1. Notation and Problem Formulation

Our notation follows that of Godfrey and Powell (2002) quite closely, with some changes required to handle the case of multiperiod travel times. Our presentation here is brief, but complete so that the paper can stand alone.

We first let:

$T =$ the number of planning periods in the planning horizon.

$\mathcal{T} = \{0, 1, \dots, T-1\} =$ the times at which decisions are made.

$\mathcal{I} =$ the set of physical locations in the network indexed by i and j.

$\tau_{ij} =$ the travel time from i to j, $i, j \in \mathcal{I}$.

$\tau_{\max} =$ the largest travel time between any pair of cities.

We define our resources using:

$\widehat{R}_{tt'} =$ the vector of resources that we first learn about at time t that will become available at time t', with element $\widehat{R}_{itt'}$ for location i;

$R_{tt'} =$ the total number of resources that will be available at time t' that we know about at time t, before any new resources are added. $R_{tt'}$ is a vector with element $R_{itt'}$;

$R^+_{tt'} =$ the total number of resources that we know about at time t that can be acted on at time $t' = R_{tt'} + \widehat{R}_{tt'}$;

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

We represent the resources as vectors because the number of different types of resources is typically not that large. By contrast, there is typically far more variety in the tasks, and we have found it more useful to represent them using sets:

$\widehat{\mathcal{L}}_{tt'} =$ the set of tasks that we first learn about in time period t that will become available at time t';

$\mathcal{L}_{tt'} =$ the set of tasks available at time t' that we know about at time t but before the new arrivals in $\widehat{\mathcal{L}}_{tt'}$ are added to the system;

$\mathcal{L}^+_{tt'} =$ the set of all tasks available to be serviced at time t' that we know about at time t, including new tasks that just arrived in this period $= \mathcal{L}_{tt'} \cup \widehat{\mathcal{L}}_{tt'}$;

$\mathcal{L}^+_t = \bigcup_{t' \geq t} \mathcal{L}^+_{tt'}$;

$b_\ell =$ the vector of attributes of task $\ell \in \mathcal{L}^+_t$. We assume that the sequence $(b_\ell)_{\ell \in \mathcal{L}}$ is drawn from a known distribution;

$\mathcal{L}^e_t =$ the tasks that *expire* at time t (either because they were served, or because they left the system for some other reason such as hitting the end of the time window), and therefore leave the system.

Throughout, we use $\mathcal{L}^+_{it}$ to represent tasks that originate at location i, and $\mathcal{L}^+_{ijt}$ to represent the set of tasks with origin i and destination j.

Let $W_t = (\widehat{R}_t, \widehat{\mathcal{L}}_t)$ represent the new information arriving in time period t. $(W_t)_{t=0}^T$ then becomes our stochastic information process, with realization $W_t(\omega) = \omega_t = (\widehat{R}_t(\omega), \widehat{\mathcal{L}}_t(\omega))$. Let $\mathcal{H}_t$ be the $\sigma-$algebra generated by $(W_0, \dots, W_t)$, with $\mathcal{H} = \mathcal{H}_T$. Then our probability space is given by $(\Omega, \mathcal{H}, \mathcal{P})$, where $\mathcal{P}$ is a probability measure defined on $(\Omega, \mathcal{H})$.

Our decisions are given by, for each $t \in \mathcal{T}$, $i, j \in \mathcal{I}$ and $l \in \mathcal{L}^+_{it}$:

$$x_{lt} = \begin{cases} 1 & \text{if a resource at } i \text{ is assigned to a task} \\ & l \in \mathcal{L}^+_{it} \text{ at time } t, \\ 0 & \text{otherwise,} \end{cases}$$

$y_{ijt} =$ the number of resources repositioned from i to j beginning at time t.

We also define the vectors $x_t \equiv (x_{lt})_{l \in \mathcal{L}^+_t}$ and $y_t \equiv (y_{ijt})_{i,j \in \mathcal{I}}$. We let the *complete* state variable be given by:

$$S^+_t = \{R^+_t, \mathcal{L}^+_t\},$$

and we define the *incomplete* state variable to be:

$$S_t = \{R_t, \mathcal{L}_t\}.$$

Below, we formulate our dynamic programming recursion in terms of the incomplete state variable.

The costs and rewards are given by:

$c_{ij} =$ the cost to reposition a resource from location i to location j;

$r_{lt} =$ the reward received for servicing task $l \in \mathcal{L}^+_t$ starting at time $t \in \mathcal{T}$.

Let $g_t(x_t, y_t)$ be the one-period reward function:

$$g_t(x_t, y_t) = \sum_{i \in \mathcal{I}} \sum_{l \in \mathcal{L}^+_{it}} r_{lt} x_{lt} - \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} c_{ij} y_{ijt}$$
$$= \text{the total profit gained from the decisions made at time } t. \quad (1)$$

Our overall objective is to solve:

$$\max_{x_0, y_0 \in \mathcal{X}_0} g_0(x_0, y_0) + E\left\{ \sum_{t \in \mathcal{T} \setminus 0} \max_{(x_t, y_t) \in \mathcal{X}_t} g_t(x_t, y_t) \right\}. \quad (2)$$

There are two types of constraints for this system. The first are the constraints that apply to the decisions made at a single point in time t under realization $\omega \in \Omega$:

$$\textstyle\sum_{j\in\mathcal{I}} y_{ijt}(\omega) + \sum_{l\in\mathcal{L}^+_{it}(\omega)} x_{lt}(\omega) = R^+_{itt}(\omega) \qquad \forall\, i \in \mathcal{I}, \tag{3}$$

$$x_{lt}(\omega) \le 1 \qquad \forall\, l \in \mathcal{L}^+_t(\omega), \tag{4}$$

$$x_{lt}(\omega) \ge 0 \qquad \forall\, l \in \mathcal{L}^+_t(\omega), \tag{5}$$

$$y_{ijt}(\omega) \ge 0 \qquad \forall\, i, j \in \mathcal{I}. \tag{6}$$

We let the set $\mathcal{X}_t(\omega)$ be the set (x_t, y_t) that satisfies Equations (3)–(6). Note that the flow conservation constraint (3) uses the complete resource vector $R^+_{itt} = R_{itt} + \widehat{R}_{itt}$ on the right-hand side. The second set of constraints control the dynamics of the system over time:

$$\begin{aligned} R_{j,t+1,t+\tau}(\omega) = &\sum_{\{i\in\mathcal{I}:\tau_{ij}=\tau\}} \left(y_{ijt}(\omega) + \sum_{l\in\mathcal{L}^+_{ijt}(\omega)} x_{lt}(\omega) \right) \\ &+ R_{j,t,t+\tau} \quad \forall\, j \in \mathcal{I}, \end{aligned} \tag{7}$$

$$\begin{aligned} \mathcal{L}_{t+1}(\omega) = \mathcal{L}^+_t(\omega) \setminus \mathcal{L}^e_t(\omega) \\ \forall\, \tau = 1, \ldots, \tau_{\max}. \end{aligned} \tag{8}$$

The functions g_t, decisions (x_t, y_t), and constraints $\mathcal{X}_t$ are all $\mathcal{H}_t$-measurable functions. Of course, we now face the problem of solving this formulation.

2. Solution Strategy

The stochastic optimization problem can be solved by first formulating the dynamic programming recursion in terms of the incomplete state variable:

$$V_t(S_t) = E\left\{ \max_{(x_t, y_t)\in\mathcal{X}_t} g_t(x_t, y_t) + V_{t+1}(S_{t+1}) \middle| S_t \right\}. \tag{9}$$

We then drop the expectation and solve the problem for a single sample realization:

$$\begin{aligned} V_t(S_t, \omega_t) = &\max_{(x_t(\omega_t), y_t(\omega_t))\in\mathcal{X}_t(S_t,\omega_t)} g_t(x_t(\omega_t), y_t(\omega_t)) \\ &+ V_{t+1}(S_{t+1}(\omega_t)). \end{aligned} \tag{10}$$

Note that by using the incomplete state variable, we may solve Equation (10) for x_t using the sample ω_t without violating the measurability conditions of the decision.

Finally, we replace the value function with a suitable approximation. We choose to write our approximation in terms of R_t instead of $S_t = (R_t, \mathcal{L}_t)$ because the vehicles are our only active resource layer. Thus, we replace $V_{t+1}(S_{t+1})$ with $\widehat{V}_{t+1}(R_{t+1})$. In the special case where tasks must be served as soon as they appear in the system, $\mathcal{L}_t$ is always empty and our approximation works extremely well. When tasks may be held, this approximation introduces errors (as we show below), and represents an opportunity to improve the quality of the solution. This step produces:

$$\begin{aligned} \widetilde{V}_t(R_t, \omega_t) = &\max_{(x_t(\omega_t), y_t(\omega_t))\in\mathcal{X}_t(R_t,\omega_t)} g_t(x_t(\omega_t), y_t(\omega_t)) \\ &+ \widehat{V}_{t+1}(R_{t+1}(\omega_t)), \end{aligned} \tag{11}$$

where $\widetilde{V}_t(R_t, \omega_t)$ serves as a placeholder. Equation (11) is solved subject to the constraints $\mathcal{X}_t(R_t, \omega_t)$. Godfrey and Powell (2002) proposed a separable approximation of the form:

$$\widehat{V}_t(R_t) = \sum_{i\in\mathcal{I}} \widehat{V}_{it}(R_{it}),$$

where the functions $\widehat{V}_{it}(R_{it})$ are estimated using the CAVE algorithm. The problem is solved by executing a forward pass through time, determining a set of decisions $(x_t)_{t\in\mathcal{T}}$ for a single sample ω, and using a particular set of value function approximations $(\widehat{V}_t)_{t\in\mathcal{T}}$. At the end of the forward pass, we use the dual variables from solving $\widetilde{V}_t$ to update the functional approximations using the CAVE algorithm.

There are two dimensions to the algorithm for multiperiod travel times: the approximation and the updating method. First, we suggest a new functional approximation that accommodates the multiperiod travel times, which is described in §2.1. Then we illustrate in §2.2 why this new approach avoids the long-haul bias associated with the single-period formulation. Finally, we propose a novel way of updating the value functions, which is given in §2.3. Although the method is an approximation (which is evaluated in §3), the technique produces near-optimal solutions when it is applied to the special case of deterministic pure networks.

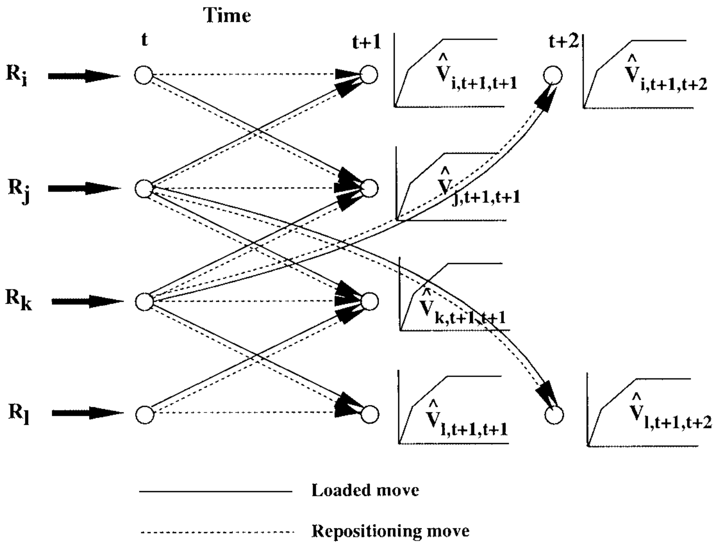

Figure 2 **Subproblem at Time *T* Expressed as a Two-Stage Network Problem**

2.1. Approximating the Value Function

Our point of departure with Godfrey and Powell (2002) is in the formulation of the approximation $\widehat{V}_t(R_t)$. In this paper, $R_t = (R_{tt'})_{t' \geq t}$. Correspondingly, we propose to express our value function as:

$$\widehat{V}_t(R_t) = \sum_{t'=t}^{t+\tau_{\max}-1} \widehat{V}_{tt'}(R_{tt'}). \tag{12}$$

The value function approximation $\widehat{V}_{tt'}(R_{tt'})$ has an economic interpretation. At time t, the history of all prior decisions is captured in the resource vector R_t. Of these resources, some are next available for assignment at time t, some at time $t+1$, and so on to time $t+\tau_{\max}-1$. The function $\widehat{V}_{tt'}(R_{tt'})$ estimates the expected value of only those resources that are next available for assignment at time t'. Since each resource is counted exactly once in the resource vector R_t, then the expected future contribution of each resource is accounted for by exactly one of the value approximations $\widehat{V}_{tt'}$. Taking the sum over all travel times gives the expected future contribution of the entire set of resources.

In this formulation, we consider only the effect of a change in the resource inventory, not the task set $\mathcal{L}_t$. Also note that although the approximation is separable in the resource vector, the original value function is not. However, the separable approximation allows us to solve the time t subproblem as a network, which is not only fast but also provides integer solutions; this is not possible with a nonseparable function.

Given the value function approximation $\widehat{V}_t(R_t)$, we generate a random outcome $\omega \in \Omega$ and solve the following sequence of network subproblems, starting with time $t = 0$ and continuing through $t = T-1$:

$$\max_{(x_t, y_t)\in\mathcal{X}_t} g_t(x_t, y_t) + \sum_{\tau=1}^{\tau_{\max}} \widehat{V}_{t+1,\,t+\tau}(R_{t+1,\,t+\tau}). \tag{13}$$

If the original problem has single-period travel times ($\tau_{\max} = 1$), then the time t subproblem (13) is equivalent to the one presented in Godfrey and Powell (2002) for single-period travel time problems.

Figure 2 illustrates the time t subproblem as a two-stage network. We refer to all the nodes representing activities at time t as the *first-stage* nodes; all the nodes representing points in time $t' > t$ are called the *second-stage* nodes. Each time t node represents a location i with corresponding supply R^+_{itt}. In the first-stage decisions (time t), each resource must: (1) serve a demand, (2) reposition, or (3) remain in inventory. From each origin node, we build three types of arcs: (1) service arcs to each demand destination; (2) repositioning arcs to each destination within an allowable radius; and (3) inventory arcs to the same location one time period later. The demand arcs have arc cost

$+r_{lt}$ and upper bound $|\mathscr{L}^+_{ijt}(\omega)|$. The repositioning arcs have cost $-c_{ij}$ and upper bound $+\infty$. The first-stage network matches that of the myopic planner who will always choose to hold inventory (cost of 0) over repositioning (cost > 0).

The second-stage decisions (time $t' = t+\tau, \tau > 0$) incorporate the value function approximation $\widehat{V}_{t+1,t'}(R_{t+1,t'})$. We further separate the approximation by location and define the new approximation $\widehat{V}_{j,t+1,t'}(R_{j,t+1,t'})$ to apply to the location j, time t' node. This new approximation simply maps a scalar (number of resources) to a scalar (future value). This enables the piecewise-linear function to be represented by a series of slopes and segment lengths represented by arcs connected to a common supersink (not illustrated in Figure 2).

For the set of arcs that describes $\widehat{V}_{j,t+1,t+\tau}(R_{j,t+1,t+\tau})$, the kth arc ($k = 0, 1, \ldots, k_{\max}$) has cost $+v^k_{j,t+1,t+\tau}$ and upper bound $u^{k+1}_{j,t+1,t+\tau} - u^k_{j,t+1,t+\tau}$ where $u^{k_{\max}+1}_{j,t+1,t+\tau} \equiv \infty$. The last arc has a finite cost and infinite upper bound to absorb overflow and to ensure that the network always has a feasible solution.

The network structure enables fast, integer optimal solutions using a network solver. In addition, we obtain dual values on all of the nodes. Of particular interest are the flow augmenting and flow decrementing dual values and paths introduced by Powell (1989) on the time t nodes. We define the dual values

$\pi^+_{i,t,t+\tau}$ = marginal value of one more resource at node i, time $t+\tau$ in the time t subproblem ($\tau = 0, 1, 2, \ldots, \tau_{\max}$);

$\pi^-_{i,t,t+\tau}$ = marginal value of one fewer resource at node i, time $t+\tau$ in the time t subproblem ($\tau = 0, 1, 2, \ldots, \tau_{\max}$).

Note that the vector of first-stage dual values $\pi^{+/-}_{tt}$ is measured relative to the initial supply R^+_{tt}. However, the second-stage dual value vectors $\pi^{+/-}_{t,t+\tau}$ ($\tau > 0$) are measured relative to the supply of resources $R_{t+1,t+\tau}$ after the first-stage actions.

We observe from Equation (7) that since each action takes *at least* one time period, $R_{t+1,t} = R^+_{t,t}$, meaning that the expected time t inventories are the same whether measured at time t or time $t+1$. With this change, we can write that the dual values $\pi^{+/-}_{i,t,t+\tau}$ are measured relative to the supply $R_{i,t+1,t+\tau}$ for $\tau = 0, 1, \ldots, \tau_{\max}$. These dual values, along with the paths that the marginal units take to the supersink, are stored for use in the updating pass.

As a final step, we update the state from S^+_t to S^+_{t+1} by implementing the first-stage time t decisions and adding new demands from the outcome ω_{t+1}. We continue generating and solving this sequence of subproblems through $t = T-1$.

2.2. A Multiperiod Example

Before describing the algorithm for updating the value function approximations, we illustrate why the single-period travel time formulation of Godfrey and Powell (2002) can provide inferior solutions when applied to problems with multiperiod travel times. We then generalize the single-period updating scheme to one that extends to multiperiod problems naturally.

In the single-period formulation, each time-space node has a single value function approximation $\widehat{V}_{it}(R_{it})$. After solving the time t subproblem, the first-stage dual values π^-_{it} and π^+_{it}, measured relative to the first-stage supply R^+_{it}, are used to update $\widehat{V}_{it}$. Given single period travel times, deciding which resources move to node i, time t (with reward function $\widehat{V}_{i,t}$) is done only in the time $t-1$ subproblem. However, this property does not hold under multiperiod travel times (Figure 3).

In this example, there are two available demands (with net profits of \$200 and \$150) at time t at location j. There are initial resource inventories of one, zero, two, and one at locations i, j, k, and l, respectively. We assume for convenience that all value function approximations are zero except for $\widehat{V}_{j,t}$, which has a slope of \$200 between zero and one, a slope of \$150 between one and two, and a slope of \$0 thereafter to represent the two available demands. The solution of the time $t-3$ subproblem is to send a resource from l to j (at a cost of \$160) to cover the \$200 demand and hold the remaining resources in inventory. The time $t-2$ subproblem solution sends a resource from i to j (at a cost of \$100) to cover the \$150 demand and holds the remaining resources. In the time $t-1$ subproblem, the two resources at k must wait in inventory because there is no remaining benefit to move to j. The total profit from this set of decisions is \$90.

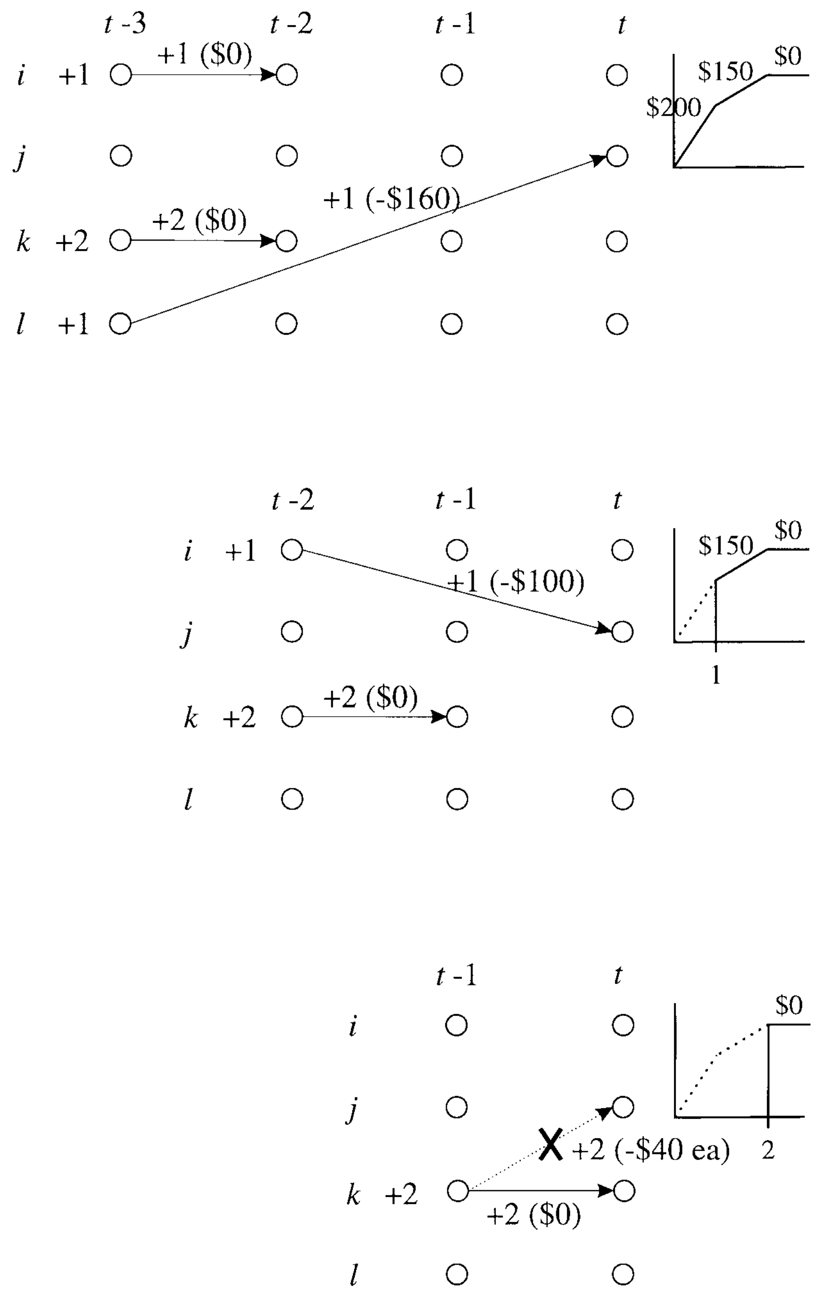

Figure 3 **Single-Period Formulation Applied to Multiperiod Travel Time Problem Gives an Inferior Solution with a $90 Profit. The Optimal Solution (Send Two Drivers from *K* to *J* at Time *T* − 1) has a $270 Profit.**

The first-stage π^- and π^+ dual values are

$$\pi^-_{..}, \pi^+_{..} = \begin{array}{c@{}c} \begin{array}{cccc} i & j & k & l \end{array} & \\ \left[\begin{array}{cccc} 0,0 & 0,0 & 0,0 & 40,0 \\ 50,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 150,0 & 0,0 & 0,0 \end{array}\right] & \begin{array}{c} t-3 \\ t-2 \\ t-1 \\ t \end{array} \end{array}.$$

For simplicity, future horizon effects beyond time t are ignored in this example. Given the single-period updating rules, there would be minor changes to $\widehat{V}_{l,t-3}$ and $\widehat{V}_{i,t-2}$ with the other approximations remaining the same. Using these new approximations, the decisions in the next iteration would remain unchanged, as would the dual values. Therefore, further iterations using the single-period updating rules would not improve this clearly inferior solution.

The inferior solution is caused by the time separation of all decisions sending flow into destination j at time t. The single-period updating rules have no mechanism for recognizing when future subproblems provide better opportunities into a particular destination/time node.

Instead, the value structure is set up such that the most valuable rewards (represented by the steepest part of the destination value function) are first made available to the resources that are the most expensive to reposition (those that are furthest away from the destination). Relatively inexpensive resources (those closest to the demand origin) can choose only from the generally less valuable portion of the value function. The order of the sequence of subproblems leads to this bias against short, inexpensive repositions. The bias would disappear if we could reverse the subproblem sequence, but this is impractical because the state updates require the forward sequence. Note that a pure linear approximation does not have this bias because the slope of the value function is constant with respect to all of the subproblems.

As suggested in Figure 3, a better solution is to replace the repositions at time $t-2$ and $t-3$ with a reposition of two resources at time $t-1$ from k to j for a net profit of \$270. In order to make more intelligent decisions under multiperiod travel times, the updating rules need to measure not just the local impact of a decision at the destination (as is done now), but the impact on intermediate time periods as well.

For the example illustrated in Figure 3, the value function $\widehat{V}_{j,t}$ measures the impact at location j, time t for all subproblems. The function has slope values \$200, \$150, and \$0 between states 0 and 1, 1 and 2, and 2 and up, respectively. For compactness, we represent this function by $\widehat{V}_{j,t} = (200, 150, 0)$. Instead of using a single value function $\widehat{V}_{j,t}$ at location j and time t, the multiperiod formulation uses a series of value functions $\widehat{V}_{j,t-\tau+1,t}$ that measures the value at location j at time t with respect to subproblem $t-\tau$ for $\tau = 1, \ldots, \tau_{\max}$.

For the example we use, the optimal value functions for the multiperiod formulation are

$$\widehat{V}_{j,t-2,t} = (40, 40, 0),$$
$$\widehat{V}_{j,t-1,t} = (40, 40, 0),$$
$$\widehat{V}_{j,t,t} = (200, 150, 0).$$

This nonintuitive result deserves explanation. The value function $\widehat{V}_{j,t,t}$ used for the time $t-1$ subproblem is as expected. Under the optimal solution, no resources are sent to location j, time t prior to the decisions at time $t-1$. At time $t-1$, two resources reposition from k to j to service the two demands at j with values \$200 and \$150.

However, the value functions used in the prior subproblems are different. At time $t-2$, there is a resource available to reposition from i to j at cost \$100. According to the value function, $\widehat{V}_{j,t-1,t}$, the value of that resource upon reaching location j, time t is \$40, not \$200 as expected. The reason is that the reposition at time $t-2$ changes the decision at time $t-1$. Only one resource repositions at time $t-1$ instead of two, thus saving \$40. This \$40 cost savings is captured in the slope of the value function $\widehat{V}_{j,t-1,t}$. The same argument applies to the value function $\widehat{V}_{j,t-2,t}$ used in the time $t-3$ subproblem.

By measuring the potential improvement instead of the absolute value of each resource, we get the desired behavior in the multiperiod formulation. If a resource from time $t-2$ or $t-3$ can arrive at location j, time t at a cost less than \$40 (by servicing a demand, for example), then that resource can be used instead of the time $t-1$ reposition. However, repositioning to location j, time t from an earlier subproblem will almost always be more expensive than the time $t-1$ reposition.

2.3. Updating the Value Function Approximation

When we solved problems with single-period travel times, it was possible to represent the process of updating the value functions using:

$$\widehat{V}_{it}^{n+1}(R_{it}^{+}) \leftarrow U^{CAVE}\big(\widehat{V}_{it}^{n}, \pi_{it}^{+}, \pi_{it}^{-}, R_{it}^{n}\big), \tag{14}$$

where U^{CAVE} represents the CAVE updating procedure. We are updating the nonlinear function using the left and right duals at the point R^n_{it}. For this problem, the duals $\pi^{+/-}_{it}$ represent estimates of the left and right gradients of subproblem (i, t) at the point R^+_{it}.

In the multiperiod travel-time problem, an additional resource into node (i, t) coming from time period $t-\tau$ may have an impact on the subproblem at time t, or any of the earlier subproblems $t-s$ for $s = 1, 2, \ldots, \tau-1$. It cannot have an impact on subproblems prior to $t-\tau$ because these decisions have already been made. If the solution to the time $t-\tau$ subproblem sends a resource into node j at time t, we use $\widehat{V}_{j,\,t-\tau+1,\,t}(R^+_{j,\,t-\tau+1,\,t})$ to estimate the value of this additional resource. The impact of this decision on the time $t-s$ subproblem for $s = 0, 1, \ldots, \tau-1$ is captured by the dual variable $\pi^+_{j,\,t-s,\,t}$. We need to estimate the net effect of moving one unit of flow from time $t-\tau$ into node j at time t. We propose to estimate this value using the formula:

$$\tilde{\pi}^+_{j,\,t-\tau,\,t} = \max_{s=0,\ldots,\tau-1} \{\pi^+_{j,\,t-s,\,t}\}. \tag{15}$$

We use the maximum value of $\pi^+_{j,\,t-s,\,t}$ for $s = 0, \ldots, \tau-1$ because this will identify the best subproblem (after $t-\tau$) for sending flow into node j at time t. Similarly, we compute the left gradient using:

$$\tilde{\pi}^-_{j,\,t-\tau,\,t} = \min_{s=0,\ldots,\tau-1} \{\pi^-_{j,\,t-s,\,t}\}. \tag{16}$$

We refer to the version of the algorithm that uses Equations (15) and (16) as the "DUALMAX" algorithm.

The dual values $\pi^-_{j,\,t-s,\,t}$ measure the local impact at time t of a decision at time $t-s$. The DUALMAX values, on the other hand, measure the impact of a decision at time $t-s$ on all future time periods up to and including time t. The DUALMAX value $\tilde{\pi}^+_{j,\,t-\tau,\,t}$ is the maximum of the future duals and measures the maximum impact to the system of adding an extra resource to destination j at time t. Conversely, the DUALMAX value $\tilde{\pi}^-_{j,\,t-\tau,\,t}$ is the minimum of the future duals and measures the minimum impact to the system of taking away a resource from destination j at time t.

Because the minimum and maximum dual values may come from different subproblems, the adjusted dual values may violate concavity ($\tilde{\pi}^-_{j,\,t-\tau,\,t} < \tilde{\pi}^+_{j,\,t-\tau,\,t}$). If this is the case, then we set $\tilde{\pi}^-_{j,\,t-\tau,\,t} = \tilde{\pi}^+_{j,\,t-\tau,\,t}$. The value function approximation $\widehat{V}_{j,\,t-\tau+1,\,t}$ is then updated using the DUALMAX values $\tilde{\pi}^-_{j,\,t-\tau,\,t}$ and $\tilde{\pi}^+_{j,\,t-\tau,\,t}$ relative to the resource state $R^+_{j,\,t-\tau+1,\,t}$. Note that we use the duals $\tilde{\pi}_{t-\tau,\,t}$ to update the function $\widehat{V}_{t-\tau+1,\,t}$.

Intuitively, we expect the DUALMAX values to produce the desired value functions because the adjustments measure the true future impact of changes to decisions in the present. Referring again to the example in Figure 3, if a resource at time $t-3$ repositions to location j, time t at a cost of \$160, then the future dual value $\pi^-_{j,\,t-1,\,t}$ measured in the time $t-1$ subproblem will be \$40, reflecting the repositioning cost from k to j. This dual value will propagate backward in time to the time $t-3$ subproblem because \$40 will be the minimum of the $\pi^-_{j,\,t-s,\,t}$ values. As a result, the value function $\widehat{V}_{j,\,t-2,\,t}$ will incorporate this future potential into its shape and the expensive reposition at time $t-3$ will be less attractive.

The second technique, which we call DUALNEXT (or NEXT), uses for $\tau = 1, 2, \ldots, \tau_{\max}$,

$$\tilde{\pi}^+_{j,\,t-\tau,\,t} = \pi^+_{j,\,t-\tau+1,\,t}, \tag{17}$$

$$\tilde{\pi}^-_{j,\,t-\tau,\,t} = \pi^-_{j,\,t-\tau+1,\,t}. \tag{18}$$

For this simpler approach, concavity will be preserved without adjustment because the underlying dual values have the desired concave property.

The intuition behind using DUALNEXT is that measuring the impact of changes to decisions in the present on the next subproblem may be sufficient. Instead of the drastic changes in decisions from iteration to iteration that may result from using the DUALMAX approach, the DUALNEXT values may change the value functions more slowly and smoothly and provide more stable results from one iteration to the next. In the limit, the necessary dual information will propagate backward in time incrementally rather than in one step.

It is important to recognize a key property of DUALMAX and DUALNEXT. If $\tau_{\max} = 1$, then this multiperiod formulation using either DUALMAX or DUALNEXT is equivalent to the single-period formulation in Godfrey and Powell (2002). Therefore, the algorithmic properties and experimental results in

```
STEP 1 Initialization:
  • Set $\nu^0_{j,t,t+\tau} = 0$ and $u^0_{j,t,t+\tau} = 0$ as the piecewise-linear approximation $\hat{V}_{j,t,t+\tau}$ for
    all $j \in \mathcal{I}, t \in \mathcal{T}, \tau = 0, \ldots, \tau_{\max} - 1$.
STEP 2 Forward Simulation:
  • Generate a random sample $\omega \in \Omega$.
  • For each time period $t = 0, 1, \ldots, T-1$,
      – Solve the network subproblem
        $\max_{x_t, y_t \in \mathcal{X}_t} g_t(x_t, y_t) + \sum_{\tau=1}^{\tau_{\max}} \hat{V}_{t+1,t+\tau}(R_{t+1,t+\tau})$
      – Store dual vectors $\pi^+_{t,t+\tau}$ and $\pi^-_{t,t+\tau}$ for all $\tau = 0, 1, \ldots, \tau_{\max}$.
STEP 3 Value Function Update using CAVE:
  • For each $t = 0, 1, \ldots, T-1$,
      – Compute the adjusted marginal contributions $\tilde{\pi}^-_{j,t,t+\tau}$ and $\tilde{\pi}^+_{j,t,t+\tau}$ for $j \in \mathcal{I}$
        and $\tau = 1, \ldots, \tau_{\max}$ using either the DUALMAX adjustment equations (15)
        and (16) or the DUALNEXT adjustment equations (17) and (18).
      – Update $\hat{V}_{j,t+1,t+\tau} \leftarrow U^{CAVE}(\hat{V}_{j,t+1,t+\tau}, \tilde{\pi}^+_{j,t,t+\tau}, \tilde{\pi}^-_{j,t,t+\tau}, R^+_{j,t+1,t+\tau})$.
  • Return to Step 2.
```

Figure 4 **The Multistage, Stochastic Value Approximation Algorithm Using CAVE for Problems with Multiperiod Travel Times**

that paper also apply to these two algorithms when applied to problems with single-period travel times.

Having explained the details of the forward simulation and value function approximation separately, we summarize the algorithm steps in Figure 4.

3. Experimental Design and Results

In this section, we extend the experimental methodology of Godfrey and Powell (2002) to include problems with multiperiod travel times. We are interested in comparing our adaptation of the multistage version of CAVE for multiperiod travel times in three dimensions. First, we compare the performance of the algorithm on deterministic problems, where we can obtain a tight bound using a linear programming package. Second, we want to evaluate the speed of the algorithm. Finally, we want to compare the algorithm on stochastic problems against a rolling-horizon procedure using deterministic forecasts.

Section 3.1 describes the experimental design. Then, §§3.2 and 3.3 describe the results of the experiments on deterministic and stochastic data sets, respectively.

3.1. Experimental Design

We conducted both deterministic and stochastic experiments. Deterministic experiments allow us to compare our solution to the tight bound provided by an LP solver, while the stochastic experiments allow us to see the true value of the approximation technique. Experiments were run which varied the number of locations, the planning horizon, and the number of resources relative to the number of tasks.

Table 1 **Parameters for Deterministic Experiments with Varied Fleet Size and Repositioning Cost**

Problem Characteristic	Attribute Value(s)
Number of locations, $\|\mathcal{J}\|$	40 locations
Planning-horizon length, T	60 periods
Number of tasks over T	3,985 tasks
Time window length	6 periods
Net revenue per loaded mile	\$1.00 per mile
Origin/destination weights	Negatively correlated
Repositioning cost per mile	\$0.80, \$1.40, \$2.00 per mile
Fleet size	200, 400, 800 resources

Table 1 gives the characteristics of the data sets used for the deterministic experiments. The runs were conducted on a problem with 40 randomly generated locations, with demands drawn from a Poisson distribution spread over the 60-period planning horizon, producing a total of 3,985 tasks. Figure 5 illustrates the distribution of travel times for the demand sample. We assume that each task may be satisfied within a time window of six periods. The rewards (\$1.00 per mile) and repositioning costs (which range from \$0.80 to \$2.00 per mile) were chosen to represent problems where repositioning is relatively cheap, to problems where it is relatively expensive. The repositioning cost is an important feature. If the cost was too high, the problem would effectively decompose by location and would become artificially easy to solve.

Tasks were generated randomly between locations, where the average number of tasks from location i to location j was proportional to $\beta_i(1-\beta_j)$ where β_i and β_j are drawn randomly from a uniform distribution between zero and one. This approach ensures that the number of tasks inbound to a location is negatively correlated with the number of tasks outbound from that location, thus forcing the system to reposition vehicles from one region to another.

3.2. Deterministic Experiments

Taking the combination of three repositioning costs and three resource sizes, we have nine different data sets to evaluate the performance of the CAVE

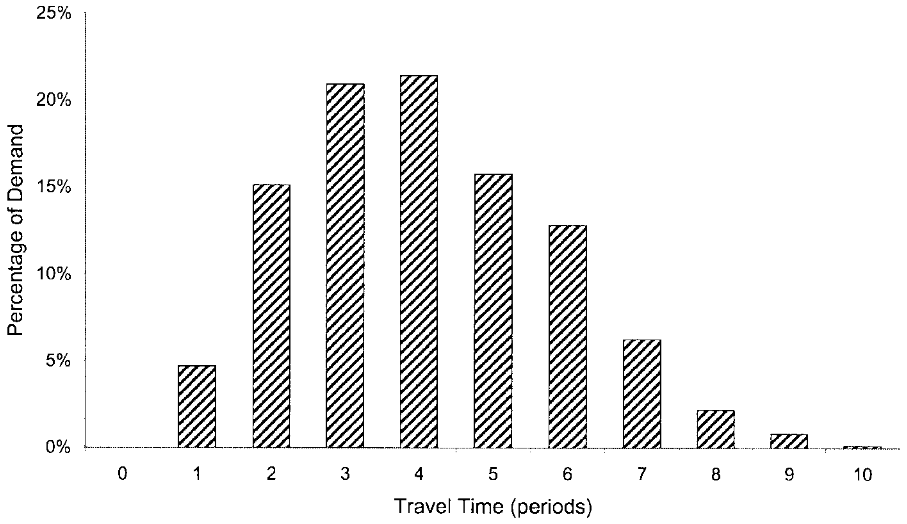

Figure 5 Distribution of Travel Times for Deterministic Task Set

logic. We apply the following six algorithms to each problem:

- CAVE using the DUALMAX dual values (Equations (16) and (15)) (denoted MAX).
- CAVE using the DUALNEXT dual values (Equations (18) and (17)) (denoted NEXT),
- CAVE using the single-period formulation (denoted Single).
- CAVE with a pure linear approximation (denoted Linear).
- The Linear Approximation and Multiplier Adjustment (LAMA) method of Carvalho and Powell (2000).
- The commercial linear programming package CPLEX.

For our first set of experiments, we ran the algorithms on problems where the tasks were fixed in time (single-period time windows). For this special case, the problems reduce to pure networks, where we can obtain optimal integer solutions, thereby providing a tight bound. This test provides a basic measure of the quality of the algorithm when applied to the simplest problem in this class. Table 2 contains the no-time window results for the DUALMAX and DUALNEXT algorithms. Across all problems, the CAVE solutions are consistently within 0.1% of optimal. In fact, the results are so good that one can reasonably pose the question of whether the algorithm may be optimal, in the limit, for this problem class. These experiments hint at the possibility of even stronger theoretical results for this algorithm.

Our remaining experiments on deterministic problems focus on examples where the tasks may be served over a period of time (time windows). Algorithms for fleet management are routinely applied to problems where the tasks are fixed in time (see, for example, Hane et al. 1995, Vemuganti et al. 1989, and Rushmeier and Kontogiorgis 1997). The extension to handle time windows is nontrivial (Rexing et al. 2000) in the context of deterministic problems. Our adaptive dynamic programming methods can handle time windows quite easily if we use the approximation $V_t(S_t) \approx \widehat{V}_t(R_t)$, which means that the decision to serve a task at time t ignores the impact of not serving the task now.

Table 2 Percentage of Integer-Optimal Value Obtained Using CAVE with No Time Windows (Network Problems)

	DUALMAX Planning Horizon			DUALNEXT Planning Horizon		
Locations	30 (%)	60 (%)	120 (%)	30 (%)	60 (%)	120 (%)
20	99.99	99.98	99.99	99.98	99.98	99.99
40	99.96	99.94	99.91	99.97	99.95	99.92
80	99.95	99.94	99.95	99.96	99.93	99.93

Table 3 Best Solution as a Percentage of the CPLEX Relaxed Solution for the First Set of Deterministic Problems

		Best Solution as a Percentage of CPLEX				
Empty Cost	Fleet Size	MAX (%)	NEXT (%)	Single (%)	Linear (%)	LAMA (%)
$0.80	200	95.88	95.80	95.64	95.09	85.03
$0.80	400	99.13	99.14	81.72	96.25	99.17
$0.80	800	99.73	99.72	73.29	94.19	99.80
$1.40	200	95.33	95.23	94.93	94.74	85.73
$1.40	400	98.81	98.80	85.93	96.10	98.73
$1.40	800	99.61	99.60	84.56	92.75	99.38
$2.00	200	94.96	94.85	94.50	94.32	85.96
$2.00	400	98.65	98.64	88.59	96.15	98.25
$2.00	800	99.55	99.56	90.68	91.40	99.30

We evaluate the algorithms based on two statistics: (1) the best objective function value after 500 iterations of CAVE and 1,000 iterations of Linear (approximately 1,500 CPU seconds each) and after 5,000 iterations of LAMA (approximately 3,600 CPU seconds) and (2) the best objective function value as a function of CPU time for each algorithm. The extra LAMA iterations are needed due to its slow convergence, as we show.

Table 3 lists the best objective function value (relative to the LP CPLEX bound) obtained by the competing algorithms as a function of the repositioning cost and the number of resources. The two variations of the multiperiod CAVE algorithm consistently produce the best results. The application of the single-period version of CAVE to a multiperiod problem works very poorly, even worse than the linear approximation. We also show comparisons against the LAMA algorithm, since this was our best algorithm for the linear approximation. It is important to keep in mind that LAMA cannot be applied to stochastic problems.

Table 4 Parameters for Deterministic Experiments with Varied Horizon Length and Number of Locations

Problem Characteristic	Attribute Value(s)
Number of locations, $\|\mathcal{I}\|$	20, 40, 80 locations
Planning-horizon length, T	30, 60, 120 periods
Number of tasks over T	Approximately 2,000, 4,000, 8,000
Time window length (fixed)	0 or 6 periods
Net revenue per loaded mile	$1.00 per mile
Origin/destination weights	Negatively correlated
Cost per repositioning mile	$1.40 per mile
Fleet size	400 resources

Figure 6 illustrates the speed of convergence for each algorithm. We see that the multiperiod CAVE algorithms converge very quickly compared to all of the competing algorithms. Note that CPLEX, for this deterministic problem, takes a while to get a feasible solution and then improves very quickly.

We next study the effect of problem size on solution quality and CPU time. The characteristics of these data sets appear in Table 4, where we vary the num-

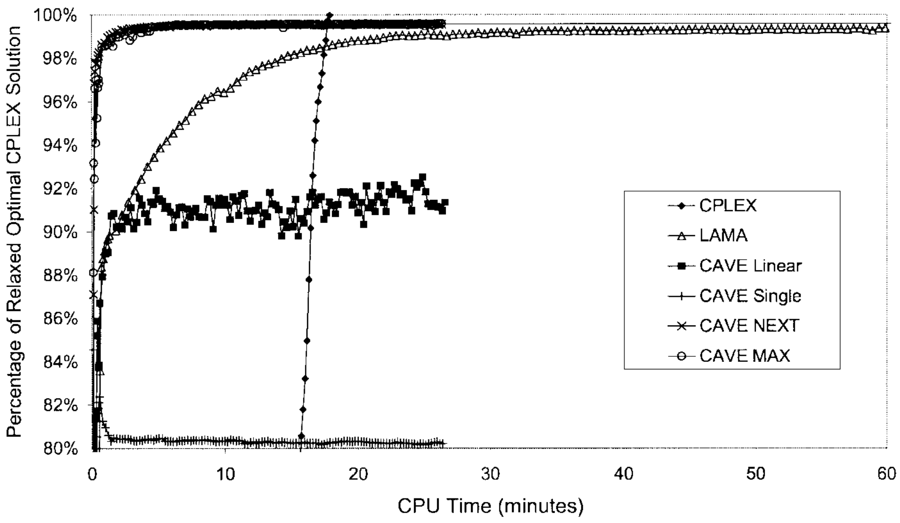

Figure 6 Solution Quality Versus CPU Time Comparison for Deterministic Problem with Time Windows (800 Drivers and $1.40 Repositioning Cost)

Table 5 Best Solution as a Percentage of the CPLEX Relaxed Solution for the Second Set of Deterministic Problems

No. of Locations	Horizon Length	Best Solution as a Percentage of CPLEX				
		MAX (%)	NEXT (%)	Single (%)	Linear (%)	LAMA (%)
20	30	99.07	99.12	89.83	94.95	99.11
20	60	99.43	99.40	88.22	96.87	98.89
20	120	99.61	99.61	87.74	97.43	98.45
40	30	98.26	98.26	86.16	94.64	98.95
40	60	98.81	98.80	85.93	96.10	98.74
40	120	99.20	99.21	85.69	96.64	98.69
80	30	96.92	96.84	78.97	93.14	98.12
80	60	97.26	97.25	76.98	93.85	97.29
80	120	98.41	98.40	78.35	95.68	95.17

ber of locations and the problem horizon. For each data set, we compare objective function values after 500 iterations of the CAVE methods (MAX, NEXT, and Single) and 1,000 iterations of CAVE Linear (comparable CPU times). For LAMA, we use 5,000, 10,000, and 15,000 iterations for 30, 60, and 120 periods, respectively. As the horizon lengthens, more iterations are required for LAMA to reach a near convergence.

Table 5 lists the best objective function value (relative to CPLEX) obtained by the CAVE methods and LAMA for each data set. In general, the multiperiod CAVE solutions are best given longer horizons and fewer locations, while LAMA has an advantage for shorter horizons and more locations. As before, the two multiperiod CAVE methods give nearly identical results.

The Single algorithm gives uniformly poor results. The reason is that the number of short repositioning opportunities grows with the number of locations. Because the Single algorithm favors long repositioning over short, these opportunities are never realized.

The most noticeable difference between the multiperiod CAVE results with tight time windows (Table 2) and the results with wider time windows is the sometimes significant gap between our results and the LP bound. While some of this gap can be attributed to the lack of an integer-optimal solution from the LP relaxation, we believe that the most important reason is the use of a value function approximation that ignores the tasks. Spivey and Powell (2000) compare results using just "resource gradients" (where $\widehat{V}_t$ is purely a function of R_t) to those which used resource and task gradients (where $\widehat{V}_t$ would be a function of both R_t and $\mathscr{L}_t$) in the context of the dynamic assignment problem. Their work shows that the use of task gradients, which capture the value of not serving a task in time t, improves overall solution quality by an average of 3 to 4%. This suggests that using task gradients for problems with time windows may significantly close the optimality gap.

Table 6 Parameters for Stochastic Experiments

Problem Characteristic	Attribute Value(s)
Number of locations, $\lvert\mathscr{I}\rvert$	20, 40, 80 locations
Number of resources	200, 400, 800 resources
Planning-horizon length, T	3, 6, 9, 15, 30, 60, 90 periods
Simulation length	180 periods
Number of tasks over simulation	Approximately 12,000
Time window length (fixed)	1 period
Net revenue per loaded mile	\$1.00 per mile
Origin/destination weights	Negatively correlated
Repositioning cost per mile	\$1.40 per mile

3.3. Stochastic Experiments

Finally, we evaluate the performance of the CAVE approximation in the presence of uncertain tasks. Table 6 gives the characteristics of the data sets used for this set of experiments. For these experiments, we assume that tasks must be served at a single point in time (tight time windows). We used this assumption to simplify the running of the rolling-horizon experiments, which require solving sequences of problems using deterministic forecasts.

The experiments were all run using a 180-period simulation where the actual demand is represented by the outcome ω^0. At time t, we observe ω_t but not the future. We determine the decisions in time t either by estimating the value function approximations using stochastic resampling over the next T time periods, or by solving a single optimization problem using a deterministic forecast of the next T time periods. When we use the value function approximations, we compare both the DUALMAX and DUALNEXT procedures for updating the functional approximation.

The rolling-horizon results that we have presented have assumed a 30-period planning horizon. Figure 7 illustrates the rolling-horizon results as a function

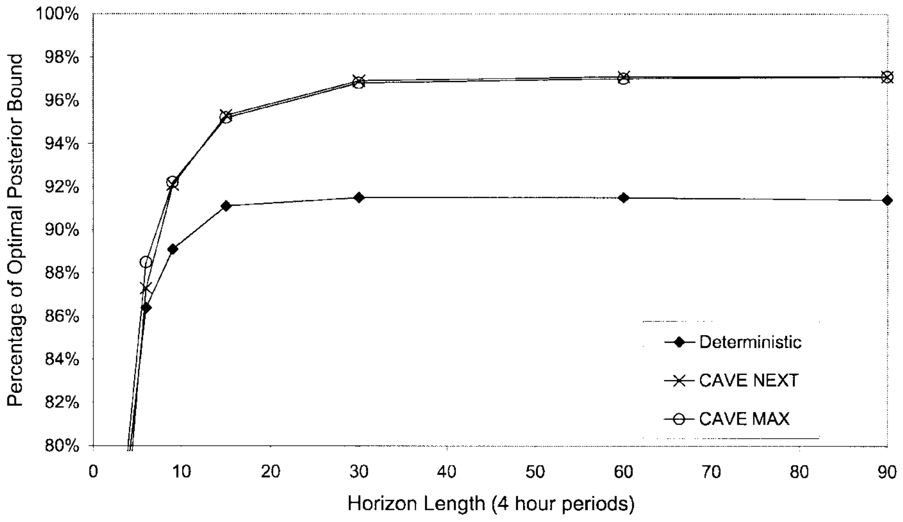

Figure 7 **Rolling-Horizon Comparison of Deterministic Solution and Stochastic Training with 40 Locations and 400 Resources**

of planning horizon length for 40 locations and 400 resources. Based on this and other experiments, we chose a planning horizon of 30 time periods for all the remaining runs.

Table 7 summarizes the results for the 30-period planning horizon length. Resampling using the CAVE methods provide a 0.5–10 % advantage over using the expectation, with greatest improvements (over 5%) for the problem with the most locations, even when there are far more resources than are required to serve the demands (the 800-resource case). The deterministic model will not, in general, allow a number of resources greater than the forecasted demand, while in practice it is often the case that the actual demand is greater than the forecast.

Comparisons against the posterior bound provide additional insights. As expected, relative solution quality drops for all categories as the number of locations increases. Also, solution quality improves for all categories as the fleet size increases. As with the deterministic experiments, the DUALMAX and DUALNEXT results are nearly identical. However, DUALNEXT appears to have a slight advantage for the smaller fleets.

Table 7 **Summary of Rolling-Horizon Results with a 30-Period Planning Horizon Using Deterministic and Stochastic Training**

		Percentage of Posterior Bound		
No. of Locations	No. of Resources	Deterministic (Expectation) (%)	Stochastic DUALMAX (%)	Stochastic DUALNEXT (%)
20	200	90.1	93.7	94.1
20	400	96.2	97.4	97.6
20	800	97.6	97.9	98.1
40	200	84.2	91.1	92.1
40	400	91.5	96.8	96.9
40	800	95.0	97.7	97.9
80	200	76.9	86.8	87.3
80	400	85.5	94.7	94.8
80	800	90.5	96.7	96.7

4. Conclusions

We have extended the single-period travel time algorithm described in Godfrey and Powell (2002) to multiperiod travel times. Two approaches, MAX and NEXT, were described for providing dual information to the value function approximations. Both approaches behave identically to the single-period formulation when applied to problems with single-period travel times. We showed experimentally that MAX and NEXT provide similar solutions over a wide range of problems.

For deterministic problems in which the entire task set is known in advance, we showed that MAX and NEXT provide consistently high-quality solutions (within 5% of a relaxed optimal bound) for a variety of fleet sizes, repositioning costs, geographic densities, and horizon lengths. In comparisons against competing algorithms, the MAX and NEXT solutions were consistently superior in final solution quality. The only exceptions were the data sets in which the LAMA method of Powell and Carvalho (2000) provided comparable or slightly better final solutions. In those cases, the MAX and NEXT approaches yielded high-quality solutions an order of magnitude or two faster than the slowly converging LAMA approach.

Finally, we estimated the value of stochastic versus deterministic modeling for stochastic problems in which the future tasks are unknown at 0.5%–10% where the realized benefit increases as the number of resources decreases. This result is consistent with the results from the single-period experiments. The MAX and NEXT results were nearly indistinguishable.

Acknowledgments

This research was supported in part by Grant AFOSR-F49620-93-1-0098 from the Air Force Office of Scientific Research and Grant DMI00-85368 from the National Science Foundation.

References

Carvalho, T. A., W. B. Powell. 2000. A multiplier adjustment method for dynamic resource-allocation problems. *Transportation Sci.* **34** 150–164.

Godfrey, G. A., W. B. Powell. 2001. An adaptive, distribution-free approximation for the newsvendor problem with censored demands, with applications to inventory and distribution problems. *Management Sci.* **47**(8) 1101–1112.

——, ——. 2002. An adaptive, dynamic programming algorithm for dynamic fleet management, I: Single period travel times. *Transportation Sci.* **36**(1).

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

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**(4) 231–243.

——, T. A. Carvalho. 1998. Dynamic control of logistics queueing network for large-scale fleet management. *Transportation Sci.* **32**(2) 90–109.

Rexing, B., C. Barnhart, T. Kniker, A. Jarrah, N. Krishnamurthy. 2000. Airline fleet assignment with time windows. *Transportation Sci.* **34**(1) 1–20.

Rushmeier, R. A., S. A. Kontogiorgis. 1997. Advances in the optimization of airline fleet assignment. *Transportation Sci.* **31** 159–169.

Spivey, M. Z., W. B. Powell. 2000. The dynamic assignment problem. Technical Report CL-00-03, Department of Operations Research and Financial Engineering, Princeton University. Princeton, NJ.

Vemuganti, R., M. Oblak, A. Aggarwal. 1989. Network models for fleet management. *Decision Sci.* **20** 182–197.

Received: December 2000; revision received: August 2001; accepted: September 2001.