JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 4, May 2007

Real-Time Planning and Control of Army UAVs Under Uncertainty

Darryl K. Ahner*
U.S. Army TRADOC Analysis Center, Monterey, USA

With advances in sensor technology and data fusion used in military operations, more information is available for decision making. A key question is how to make effective use of this information. Higher level sensors cue lower level sensors, in this case unmanned aerial vehicles (UAVs), to indicate potential target arrivals. Given probability distributions of these target arrivals, simulation and mathematical programming are used within a dynamic programming framework to determine control strategies for UAVs. An adaptive dynamic programming methodology is presented for the a uniform travel time UAV planning and control problem. Special structure of a network assignment problem is exploited to recursively update functional approximations representing future rewards through the network assignment problem's subgradient information. We develop an approximate dynamic approach to real-time planning and control of unmanned aerial vehicles with a focus on accounting for stochastic arrivals of new tasks. Experimentation demonstrates the use of this method and its potential for providing quick real-time controls for UAVs. Approaching the UAV routing problem using Adaptive Dynamic Programming offers a tractable framework in which to solve these difficult problems.

Nomenclature

b_t^k	Network supply/demand vector at time t iteration k
$C_t(S_t, u_t)$	Reward at time t given S_t and u_t
D_i^-	negative directional derivative by decrementing amount at node i
D_i^+	positive directional derivative by incrementing amount at node i
$f_1(), f_2()$	Functions for state dynamics
h_t	History of information at time t
I_t^L	indicator function equal to 1 if task has not yet arrived and 0 otherwise
$J_t()$	Value-to-go function
$J_t^u()$	Value-to-go function
$\mathcal{J}$	set of physical locations/nodes in the network
$\mathcal{J}_t$	set of physical locations/nodes in the network where vehicles are located at time t
$\mathcal{J}_{t+1}$	set of physical locations/nodes in the network indexed by i and j where vehicles can move to at time $t+1$ given $\mathcal{J}_t$
$\tilde{J}^{n,u_t}()$	Approximation of value-to-go function at iteration n

Received 17 August 2006; revision received 8 February 2007; accepted for publication 8 February 2007.
* Analyst, TRADOC Analysis Center-Monterey, Naval Postgraduate School, P.O.Box 8695, Monterey, CA 93943, and AIAA Member. darryl.ahner@us.army.mil.

L_t	vector of tasks available at time t but before new arrivals in $\hat{L}_t$ are added to the system
L_t^e	tasks that expire at time t because they were served
L_t^+	set of all tasks available to be serviced at time t
$\hat{L}_t$	vector of tasks that arrive at time t
$\hat{L}_{it}$	number of tasks that first become available at location i at time t
N_t^k	arc-incidence matrix at time t iteration k
r_{it}	the reward received at location i by vehicle k at time t
R_t	vector of vehicles over the set $\mathcal{J}_t$
R_{it}	number of vehicles available at location i at time t
S	State space of the system
$S_t \in S$	State at time t
S_t^u	Post-decision state at time t
$s' \in S$	A given state in the state space
T	number of planning periods in the planning horizon
$\mathcal{T} = \{0, 1, \ldots, T\}$	times at which decisions are made
u_t	Decision at time t
v_{it}^n	Subgradient estimates
W^n	Exogenous information sample for all time periods
W_t	Exogenous information first known at time t
Y_j	Random variable for task arrival j
α	Smoothing factor
π_{it}^n	the dual values at iteration n and time t for node i
$\tilde{v}_{it}^k$	estimate marginal value of one more resource at node i at time t iteration k
ξ_t	Linear functional estimate at time t
Subscript	
t	Time

I. Introduction

WITH advances in sensor technology and data fusion used in military operations, more information is available for decision making. A key question is how to make effective use of this information. One specific area the Army has funded is the development of Unmanned Aerial Vehicles (UAVs) as part of its Future Combat Systems (FCS). These UAVs use information from higher level sensors as cues. With this information probability distributions can be developed to represent the possibilities of new enemy targets within a military units area of operations.

In this paper, we consider routing and scheduling of UAVs in a dynamic stochastic environment motivated by surveillance operations. Unlike standard routing and scheduling problems, the problems associated with UAVs in surveillance operations involve uncertain effects such as the risk of UAV loss and arrival of new tasks. In addition to addressing these uncertain effects, the decisions must be made in real time. This research provides decision support to military systems under development such as the Future Combat System (FCS) through planning and control of UAVs that perform the reconnaissance, surveillance, and target acquisition (RSTA) mission. FCS is the proposed approach of conducting future Army ground operations, integrating automated and human elements. UAVs and their efficient use are critical in this approach. UAVs perform the dirty, dull, and dangerous surveillance missions that enable Army ground operations to have less armor and to be more mobile.

Previous UAV research on planning and control of UAVs in Air Force operations use techniques[5,14] which are similar to vehicle routing problems, in that, an entire plan can be constructed from the start of the mission to completion of the mission and computation can take from one to several hours without impacting the mission. Targets or areas of interest tend to be less mobile. While replanning may occur if new information is received, replanning is not always required. When replanning is required in Air Force operations, the amount of time to replan allows planning of a complete schedule using known vehicle routing techniques. In contrast, Army tactical operations are more dynamic and the value of planning missions from their beginning to completion is much less useful because the state of the system is constantly changing, sometimes in unpredictable ways. Tasks may move, new tasks may arrive, tasks may

leave without being serviced, and UAVs may be lost. These changes are compounded by the limited fuel and limited time on station of Army UAVs. Decisions must be made within a maximum of a few minutes to account for the rapidly changing state of the battlefield and preserve fuel. In contrast to previous work on stochastic scheduling, the state is much more dynamic and UAVs may have a short term plan for the next couple of locations to visit or tasks to be completed. The plan changes often and must be updated in near real-time. Such problems can be formulated as dynamic scheduling problems under uncertainty, and can be solved in principle by stochastic dynamic programming techniques. However, due to the size and complexity of the state space in these problems, dynamic programming becomes intractable. In this paper, we develop an approximate dynamic programming approach using forms of model predictive control. In model predictive control, current control actions are determined at each time by solving a finite horizon control formulation based on the current state. As new information is acquired, the problems are reformulated and solved to obtain revised controls. While model predictive control has developed substantially since its introduction in the late seventies, this appears to be the first application of model predictive control for UAV planning and control under uncertainty and is clearly the first time that adaptive dynamic programming has been applied to the UAV planning problem.

The goal of this approach is to develop new computationally feasible and near-optimal scheduling solutions for UAVs operating in surveillance operations where adaptations can be made in real time to new information. The approach addressed here includes models with uncertain task arrivals; we develop a new formulation based on models for package pickup and delivery problems. Adaptive dynamic programming uses an "approximate" future reward function within the dynamic programming framework. Often this approximate reward function is constructed through simulation. In this paper, we develop a class of simulation-based algorithms that iteratively learn piece-wise linear cost-to-go approximations that can be used in adaptive dynamic programming to generate fast optimal strategies. Specifically, we develop these algorithms using adaptive dynamic programming using a post-decision state. It is these fast strategies that lend themselves to real-time applications.

II. Adaptive Dynamic Programming and Dynamic Programming Using a Post-Decision State

In this section we develop a problem formulation to consider probabilistic models of future tasks arrivals. To address this we develop a stochastic programming formulation that allows for real time observation of new task arrivals, and develop a new class of solution algorithms based on approximation techniques known as adaptive dynamic programming.[12]

Adaptive dynamic programming has grown out of a series of articles involving solutions of multistage stochastic programming problems. It has been used in diverse applications such as: vehicle dispatching,[9] multi-product lot sizing,[10] fleet management,[7] and the dynamic assignment problem.[15] Adaptive dynamic programming mitigates the computational problems of stochastic dynamic programming: it deals with large state spaces through value function approximation, large outcome spaces through Monte Carlo sampling, and large action spaces through solution of network problems within the dynamic programming recursion.

We develop the post-decision state dynamic programming recursion and prove its equivalence to the well-known pre-decision dynamic programming recursion.[3] We use this post-decision state dynamic programming recursion to develop an adaptive dynamic programming algorithm as presented by Powell.[7] Finally, we formulate a one-step travel time UAV routing problem with stochastic target arrivals as a post-decision state dynamic programming problem and develop the techniques needed to solve this multistage stochastic programming problem.

Consider a finite space and discrete time horizon dynamic programming problem. Let S be the state space of the system. The finite time horizon is $t = 0, \ldots, T$. The state $S_t \in S$ represents the state at time $t = 0, \ldots, T$. A decision u_t that acts on the system is selected from a finite set U at each time step. The state evolves according to a state equation which has the form

$$S_{t+1} = f_1(S_t, u_t, W_t) \tag{1}$$

where f_1 is a function describing the system dynamics, S_t includes the tasks yet to be done and UAV location, and W_t are the new task arrivals up to time t. The decision variable u determines which UAV will be assigned to a task or potential task if the task arrival is a random event. The dimensionality of u consists of the number of UAVs times the number of tasks or potential tasks.

A policy is a mapping $\Pi(S_t) : S_t \rightarrow U$ that determines a decision as a function of the state, i.e. $u_t = \Pi(S_t)$. We represent the one-period contribution to the reward as $C_t(S_t, u_t)$. We express the T-stage value to be maximized as the expected value of the summation of the T costs:

$$\max_{\pi \in \Pi(S_t)} \mathbb{E} \left\{ \sum_{t=0}^{T} C_t(S_t, u_t) | S_0 \right\} \tag{2}$$

Looking at this stochastic problem, new task arrival information becomes available after the decisions at time t, u_t, are made. This exogenous information, W_t, becomes available before the state variable S_{t+1} is measured. Therefore, the history of information is

$$h_t = (S_0, u_0, W_0, S_1, u_1, W_1, \ldots, u_{t-1}, W_{t-1}, S_t) \tag{3}$$

Let $C_t(S_t, u_t)$ be the contribution received in period t given the state S_t and decision u_t. The information that arrives time t is W_t, a random variable generated with a known probability distribution. It is well known that problems of the form given in equation (2) can be solved by Bellman's optimality equations:[2]

$$J_t(S_t) = \max_{u_t} \mathbb{E}_t \{ C_t(S_t, u_t) + J_{t+1}(S_{t+1}(S_t, u_t, W_t)) | S_t \} \tag{4}$$

$J_t(S_t)$ is commonly referred to as the value-to-go. Equation 4 requires us to take into account the impact of current decisions on future decisions which depend on future policies. As shown in [16], given Ω_t is the outcome space at time t, we can write the expectation as

$$\mathbb{E}_t \{ J_{t+1}(S_{t+1}(S_t, u_t, W_t)) | S_t \} = \sum_{s' \in S} \sum_{W_t \in \Omega_t} P(W_t) 1_{s' = f_1(S_t, u_t, W_t)} J_{t+1}(s') \tag{5}$$

$$= \max_{u_t} C_t(S_t, u_t) + \sum_{s' \in S_{t+1}} p(s' | S_t, u_t) J_{t+1}(s') \tag{6}$$

where S is the set of potential states. The one-step transition matrix letting $p(s'|S_t, u_t) =$ the probability the system will be in state s' given that the current state is S_t and we take action u_t. Then our optimality equations are:

$$J_t(S_t) = \max_{u_t} C_t(S_t, u_t) + \sum_{s' \in S_{t+1}} p(s' | S_t, u_t) J_{t+1}(s') \tag{7}$$

Therefore, if $J_{t+1}(s')$ were known, we would have to sum over the entire outcome space followed by the maximization over u_t.

As an alternative, consider a post-decision state S_t^u at time t. The post decision state evolves according to a state equation which has the form

$$S_t^u = f_2(S_t, u_t) \tag{8}$$

where f_2 is a function describing the post-decision state dynamics. This post-decision state is the state that is acted upon by a decision u_t but has not yet been acted upon by the exogenous information process, W_t. The post-decision state is therefore a function of the state at time t, S_t, and the decision at time t, u_t. We rewrite our his history of information as

$$h_t = (S_0, u_0, S_0^u, W_0, S_1, u_1, S_1^u, W_1, \ldots, u_{t-1}, S_{t-1}^u, W_{t-1}, S_t)$$

In,[13] noting that S_{t+1} is a function of S_t^u and W_t, it is shown that if the dynamic programming recursion is written around the post-decision state we obtain the post-decision version of Bellman's equation:

$$J_t^u(S_t^u) = \mathbb{E}_t \{ \max_{u_{t+1}} C_{t+1}(S_t^u, W_t, u_{t+1}) + J_{t+1}^u(S_{t+1}^u) | S_t^u \} \tag{9}$$

where the value function is indexed with the superscript u to denote the calculation from a post-decision state. Note that writing the function in terms of the post-decision state allows the expectation to move outside the maximum

operator which is a property that we will exploit later. Since the expectation is conditioned on S_t^u, the information W_t is needed in order to compute u_{t+1}.

The post-decision state variable requires the solution of a maximization problem within an expectation. The decision function is now given by

$$u_{t+1} = \arg\max_{u_{t+1}} C_{t+1}(S_t^u, W_t, u_{t+1}) + J_{t+1}^u(S_{t+1}^u(S_{t+1}, u_{t+1})) \tag{10}$$

Define the transition functions

$$S_t = f_3(S_{t-1}^u, W_{t-1})$$
$$S_t^u = f_2(S_t, u_t)$$

Then the optimality equations for the post-decision state variable are given by

$$J_t^u(S_t^u) = \mathbb{E}_t\{\max_{u_{t+1}} C_{t+1}(S_t^u, W_t, u_{t+1}) + J_{t+1}^u(f_2(f_3(S_t^u, W_t), u_{t+1}))|S_t^u\} \tag{11}$$

For notational convenience use S_{t+1}^u in place of $f_2(f_3(S_t^u, W_t), u_{t+1})$ with the realization that it is a function of (S_t^u, W_t, u_{t+1}). We refer to the set of equations given by Equation (11) as the post-decision optimality equations.

In principle, an optimal policy is found by first numerically solving Bellman's equation and then computing the optimal policy using the resulting value function. However, this requires computation and storage of $J_t^u(S_t^u)$ for each post-decision state which is generally not feasible for combinatorial problems.

In the following theorem, we show the equivalency of the post-decision Bellman equations with the more traditional Bellman equations.

Theorem 1. *The post-decision Bellman equation is equivalent to the Bellman equation in traditional dynamic programming. Given*

$$J_t(S_t) = \max_{u_t} \mathbb{E}_t\{C_t(S_t, u_t) + J_{t+1}(S_{t+1})|S_t\} \tag{12}$$

and

$$J_t^u(S_t^u) = \mathbb{E}_t\{\max_{u_{t+1}} C_{t+1}(S_t^u, W_t, u_{t+1}) + J_{t+1}^u(S_{t+1}^u)|S_t^u\} \tag{13}$$

Then

$$J_t^u(S_t^u) = \mathbb{E}_t\{J_{t+1}(S_{t+1})|S_t^u\} \tag{14}$$

Proof: As shown in [1] the optimal T-stage value is generated after T iterations of Bellman's equation:

$$J_t(S_t) = \max_{u_t} \mathbb{E}_t\{C_t(S_t, u_t) + J_{t+1}(S_{t+1}(S_t, u_t, W_t))|S_t\}$$

We note that $S_t = f_3(S_{t-1}^u, W_{t-1})$ and substitute

$$J_t(S_{t-1}^u, W_{t-1}) = \max_{u_t} \mathbb{E}_t\{C_t(S_{t-1}^u, W_{t-1}, u_t) + J_{t+1}(S_t^u, W_t)|S_{t-1}^u, W_{t-1}\}$$

$$J_t(S_{t-1}^u, W_{t-1}) = \max_{u_t} C_t(S_{t-1}^u, W_{t-1}, u_t) + \mathbb{E}_t\{J_{t+1}(S_t^u, W_t)|S_{t-1}^u, W_{t-1}\}$$

Note that $S_t = f_3(S_{t-1}^u, W_{t-1})$ and $S_t^u = f_2(S_t, u_t)$. Thus we can add S_t^u to the conditioning.

$$J_t(S_{t-1}^u, W_{t-1}) = \max_{u_t} C_t(S_{t-1}^u, W_{t-1}, u_t) + \mathbb{E}_t\{J_{t+1}(S_t^u(S_{t-1}^u, u_t, W_{t-1}), W_t)|S_{t-1}^u, W_{t-1}, S_t^u\}$$

Letting $S_t^u = f_2(S_t, u_t)$ the equation reduces to

$$J_t(S_{t-1}^u, W_{t-1}) = \max_{u_t} C_t(S_{t-1}^u, W_{t-1}, u_t) + \mathbb{E}_t\{J_{t+1}(S_t^u(S_{t-1}^u, u_t, W_{t-1}), W_t) | f_2(S_t, u_t)\}$$

Taking the conditional expectation of both sides with respect to S_{t-1}^u

$$\mathbb{E}_t\{J_t(S_{t-1}^u, W_{t-1}) | S_{t-1}^u\} = \mathbb{E}_t\{\max_{u_t} C_t(S_{t-1}^u, W_{t-1}, u_t) + \mathbb{E}_t\{J_{t+1}(S_t^u(S_{t-1}^u, u_t, W_{t-1}), W_t) | f_2(S_t, u_t)\} | S_{t-1}^u\}$$

Defining $J_t^u(S_t^u) = \mathbb{E}_t\{J_{t+1}(S_t^u, W_t) | f_2(S_t, u_t)\}$ and substituting yields

$$J_{t-1}^u(S_{t-1}^u) = \mathbb{E}_t\{\max_{u_t} C_t(S_{t-1}^u, W_{t-1}, u_t) + J_t^u(S_t^u) | S_{t-1}^u\}$$

Therefore, the pre-decision and post-decision optimality equations are equivalent. ■

The equivalency of the post-decision Bellman equations with the more traditional Bellman equations is important because the optimality of the pre-decision optimality equations is proven.[2,16] The equivalency of the post-decision optimality equations also result in an optimal solution if the exact function $J_t^u(S_t^u)$ is known.

III. The General Adaptive Dynamic Programming Algorithm

Godfrey and Powell[7] introduce an adaptive dynamic programming algorithm for stochastic dynamic resource allocation problems. Consider the general problem as defined in section II where the new task arrival information, W_t, is independent of the decision vector u_t and the state S_t. Furthermore, let these arrivals represent the arrival of tasks to the system. Assume we have an approximation of the value-to-go function $\tilde{J}_t^{n-1}(S_t^u)$ where n is the current iteration. Given a realization $\{\tilde{W}_0, \tilde{W}_1, \ldots, \tilde{W}_{T-1}\}$, we compute u_0 according to

$$u_0 = \arg\max_{u_0 \in \mathcal{U}_0} C_0(S_0, u_0)) + \tilde{J}_1^{u,n-1}(S_0^u(u_0)) \tag{15}$$

we can use a forward algorithm recursively from time $t = 1, \ldots, T-1$ to compute $S_1, u_1, \ldots$ according to

$$u_t = \arg\max_{u_t \in \mathcal{U}_t} C_t(S_{t-1}^u, \tilde{W}_{t-1}, u_t)) + \tilde{J}_{t+1}^{u,n-1}(S_t^u(u_t)) \tag{16}$$

where $\mathcal{U}_t$ represents the feasible control space given by the state dynamics. This is possible since given $\{\tilde{W}_0, \tilde{W}_1, \ldots, \tilde{W}_{T-1}\}$ all information is known to solve equation 16 as a deterministic problem which is then used to update the approximation $\tilde{J}_{t+1}^{u,n-1}(S_t^u(u_t))$ for the next sample of $\{\tilde{W}_0, \tilde{W}_1, \ldots, \tilde{W}_{T-1}\}$.

We can use the sequence of decisions selected for a given sample path of task arrivals, the associated reward values, and subgradient information to update our approximation of the value-to-go function. We outline the algorithm below:

We call the following algorithm the general adaptive dynamic programming algorithm because it only shows the overall procedure and not the specifics of how $\tilde{J}_t^{n-1}(S_t^u)$ is updated which is problem dependent.

Step 0 Initialization: Initialize $\tilde{J}_t^0$, $t = \{0, \ldots, T\}$, Set n=0

Step 1 Do while $n \leq N$: Choose $W^n \in \Omega$ where W^n is a single sample realization of task arrivals.

Step 2 Do for $t = 0, 1, \ldots, T-1$:

2a Solve

$$u_t = \arg\max_{u_t \in \mathcal{U}_t} C_t(S_{t-1}^u, \omega_{t-1}, u_t)) + \tilde{J}_{t+1}^{u,n-1}(S_{t+1}(u_t))$$

2b Update the state $S_t = f_3(S_{t-1}^u, \omega_{t-1})$.

2c Update the value function approximation $\tilde{J}_t^{n-1}(S_t^u)$ using information collected from the optimization in step 2a (Defined in section V subsection A for the uniform travel time problem. Update is through eqn (55).)

Step 3 Return $\tilde{J}^{u,N}$ and use it to select u_0

This algorithmic approach is similar to approximate dynamic programming as shown by Bertsekas and Tsitsiklis.[4] Our approach differs in two main ways. First, the post-decision state and a single Monte Carlo sample are used rather than sampling a number of forward trajectories as part of an updating process. Second, the approximate value function is updated after each Monte Carlo sample.

The general adaptive dynamic programming algorithm must be tailored for the traits of a specific problem. Important issues are the form of the value-to-go approximation, the solution method used to solve the recursion in equation (16), and the methods used to update the value-to-go approximation. In the next sections we explore use of adaptive dynamic programming for problems that relate to the routing of UAVs.

IV. Adaptive Dynamic Programming With Uniform Travel Time

In this section, we consider vehicles that travel between locations and service tasks. These tasks may be already at the location or may arrive at some future time characterized by a probability distribution function. We consider the system that evolves in discrete time over T periods. The main assumption that we make is that the travel time between two locations connected by an arc is the same for all arcs. This assumption is not unreasonable when considering that the problem is defined in discrete time. Therefore the Cartesian space can divided into hexagons in order to simplify the problem so that the UAVs travel a uniform distance from node to node.

To illustrate the problem of interest consider the example in Fig. 1. The travel time of a vehicle between any two nodes along an arc is one time unit. The state of the system at time t consists of the node location of all vehicles, the number of travel units remaining for each vehicle, the location and status (arrived or not arrived) of any remaining targets and the value of all targets. Additional targets may arrive according to some probability distribution. A vehicle uses one move per time unit whether it moves or not. Vehicles can move only to locations that are connected to their current location. The decision at each time step is whether to move or not and, if the vehicle moves, to what adjacent location. Reward is received by colocating a vehicle with a task, whereupon that task is removed and the reward is collected. The objective is to maximize the sum of the rewards collected by all vehicles. There is a finite number of vehicles, each with a finite number of allowed moves.

Next, we define the notation and problem dynamics for modeling the uniform travel time problem. The notation is adopted, in general, from the notation suggested in.[7] The physical and temporal elements of the uniform travel time problem are defined as follows:

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

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

$\mathcal{J}$ = the set of physical locations/nodes in the network.

$\mathcal{J}_t$ = the set of physical locations/nodes in the network where vehicles are located at time t.

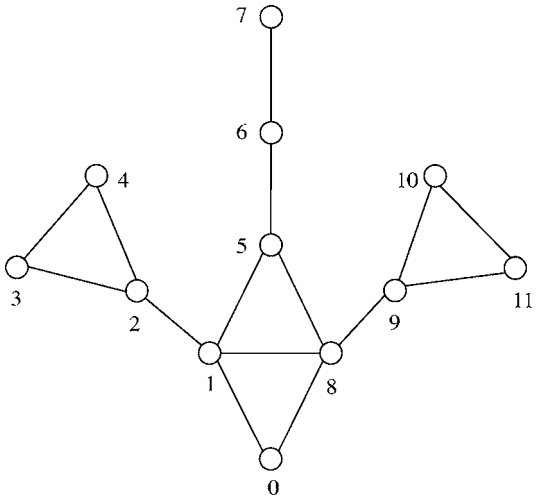

Fig. 1 Another example of the one-step problem.

$\mathcal{J}_{t+1}$ = the set of physical locations/nodes in the network indexed by i and j where vehicles can move to at time $t+1$ given $\mathcal{J}_t$.

There is a total of K vehicles in the system along with M potential tasks. Each location has either one task present or a maximum of one task arriving but not both. Therefore, the K resources are never co-located so that each element of the vectors defined below is either 0 or 1.

The resources are defined as:

R_{it} = the number of vehicles available at location i at time t. In uniform travel time problem R_{it} takes on the values of 0 or 1.

R_t = the vector of vehicles over the set $\mathcal{J}_t$.

Vehicles are represented as vectors since the number of vehicles is not typically large. Resources in[7] can have random times of arrival in the dynamic fleet management problem. In contrast to,[7] no new vehicles enter the system in this formulation of the uniform travel time problem. The number of tasks, however, can vary greatly in the system. Tasks are defined as:

$\hat{L}_{it}$ = the number of tasks that first become available at location i at time t. In uniform travel time problem $\hat{L}_{it}$ takes on the values of 0 or 1.

$\hat{L}_t$ = the vector of tasks that arrive at time t.

L_t = the vector of tasks available at time t but before new arrivals in $\hat{L}_t$ are added to the system.

L_t^+ = the set of all tasks available to be serviced at time t including new tasks that just arrived in this period. $L_t^+ = \hat{L}_t + L_t$.

L_t^e = the tasks that expire at time t because they were served.

I_t^L = indicator function equal to 1 if task has not yet arrived and 0 otherwise.

Define a random variable $\hat{L}_j(t)$ for each task arrival at location j at time t according to:

$$\hat{L}_j(t) = \begin{cases} 1 & \text{if task } j \text{ has arrived} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

The arrivals of tasks are defined by a Markov chain as:

$$P(\hat{L}_j(t) = 1 | \hat{L}_j(t-1) = 1) = 1 \tag{18}$$

$$P(\hat{L}_j(t) = 1 | \hat{L}_j(t-1) = 0) = p_j \tag{19}$$

$$P(\hat{L}_j(t) = 0 | \hat{L}_j(t-1) = 0) = 1 - p_j \tag{20}$$

These variables are mutually independent and the Markov property gives independence of the future evolution given the present state.

Let $W_t = \hat{L}_t$ represent the new tasks or information arriving in time period t. $(W_t)_{t=0}^T$ then becomes our stochastic information process, with the realization $W_t(\omega) = \omega_t = \hat{L}_t(\omega)$

Our decisions are given by, for each $t \in \mathcal{T}$ and $i, j \in \mathcal{J}$:

$$u_{ijt} = \begin{cases} 1 & \text{if a vehicle is assigned from location } i \text{ at time } t \text{ to location } j \text{ at time } t+1 \\ 0 & \text{otherwise} \end{cases}$$

Define the costs as:

r_{it} = the reward received at location i by vehicle k at time t when it leaves location i and there was a task present at location i at time t. Note that the reward is received upon leaving a location that has a task present. $r_{it} = 0$ is no task is present.

Let $g_t(u_t)$ be the one-period reward function:

$$\begin{aligned} g_t(u_t) &= \sum_{i\in\{\mathcal{J}_t\}} \sum_{j\in\{\mathcal{J}_{t+1}\}} r_{it} u_{ijt} \\ &= r_t^T u_t \\ &= \text{the total profit gained from decisions made at time } t \end{aligned} \tag{21}$$

where u_t is the specially formed vector accounting for the decision to move from node $i \in \{\mathcal{J}_t\}$ to all adjacent nodes $j \in \{\mathcal{J}_{t+1}\}$ and r_t is the corresponding reward vector.

Therefore, in this stochastic system the total reward to be maximized is

$$\max_{u_t\in\mathcal{U}_t} \mathbb{E}\left\{\sum_{t=0}^{T} g_t(u_t)\right\} \tag{22}$$

subject to the dynamics of the system:

$$R_{j,t+1} = \sum_{i\in\mathcal{J}_t} u_{ijt},\ \forall j \in \mathcal{J}, \tag{23}$$

$$L_{t+1}(\omega) = L_t^+(\omega) - L_t^e, \tag{24}$$

If only one vehicle can act on a task the following additional constraint is considered:

$$R_{jt} \le 1, \quad \forall i \in \mathcal{J}_{t,j} \in \mathcal{J}_{t+1} \tag{25}$$

The dynamics of the system are shown in Fig. 2.

The change that the decision induces on the tasks available can be represented by $L_t^+ = L_t - L_t^e$ if no tasks expire if not serviced. This is the set of tasks at time t after the decisions at time t have been carried out but before the arrival of new tasks at time t. Similarly, the change that the decision induces on the vehicles' locations can be represented by $R_t^u = R_{t+1}$ because no new vehicles arrive at time t. Therefore, we define a post decision state:

$$S_t^u = \{R_t^u, L_t^+, I_t^L\}$$

whereas the state of the decision at time $t+1$, S_{t+1} is typically defined in dynamic programming as

$$S_{t+1} = \{R_{t+1}, L_{t+1}, I_t^L\}$$

which includes the effects from the dynamics of the decision made at time t on the vehicles' locations plus the new arrivals $\hat{L}_t$ at time t.

Referring to the dynamics shown in Fig. 2, the problem begins with as initial state $S_0 = (R_0, S_0)$ from which a decision u_0 is made. This decision is implemented and the post-decision state S_0^u results. A reward g_0 is gained based upon the decision made. New tasks $\hat{L}_0$ arrive according to the Markov chain resulting in L_0^+. At time $t = 1$ the decisions result in R_1 because vehicles have moved but no tasks have entered or left the system. The vector of tasks $L_1 = L_0^+ - L_0^e$ because L_0^+ accounts for the new tasks that have arrived and L_0^e are the tasks that have been serviced in the previous time period. Note that $g_0 = 0$ if no tasks are colocated with vehicles at time $t = 0$ and that g_T is the value for the tasks that are colocated with vehicles at the last time step.

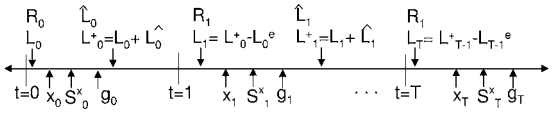

Fig. 2 Dynamics of the one-step problem.

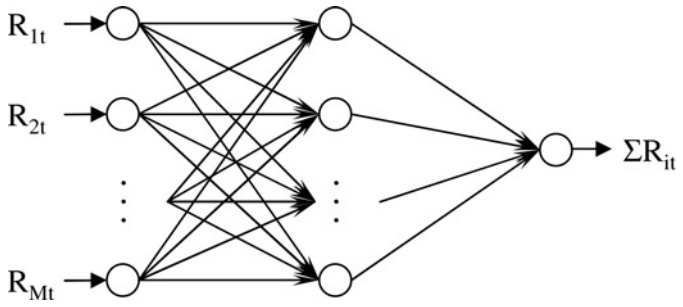

Fig. 3 Network assignment problem.

The optimality equations for this problem are given by

$$J^u_{t-1}(S^u_{t-1}) = \mathbb{E}_t\{\max_{u_t \in \mathcal{U}_t} r_t u_t + J^u_t(S^u_t)|S^u_{t-1}\} \tag{26}$$

The future value-to-go function $\mathbb{E}_t\{J^u_t(S^u_t)\}$ is unknown. As we did for the SWTA problem we will adaptively estimate an approximate value function, $\tilde{J}^u_t(S^u_t)$ so we solve the following formulation

$$J^u_{t-1}(S^u_{t-1}) = \mathbb{E}_t\{\max_{u_t \in \mathcal{U}_t} r_t u_t + \tilde{J}^u_t(S^u_t)|S^u_{t-1}\} \tag{27}$$

The approximate value function must be chosen so that it is simple enough to perform the calculations and update quickly but that it also approximates the true value-to-go function values well.

We form the problem in equation (27) as a network problem. We refer to this problem as the primal problem from which we will later obtain dual variables. The network problem is illustrated in Fig. 3. The network problem is specially formed so that every node at which a UAV could be located is represented as a source node. The arcs allow a UAV at a source node to be assigned to any adjacent node. While the network problem at first appears to be an overly convoluted formulation, the special structure of this formulation, specifically the dual information gained by the constraints, is used in the following sections to develop a simulation-based algorithm that iteratively learns cost-to-go approximation information that can be used in adaptive dynamic programming to generate fast optimal strategies.

The problem formulation for a given dynamic programming problem recursion at any time t using the linear approximation $\tilde{J}^u_{t+1}(S^u_{t+1}) = \xi_{t+1}u^k_t$ is given as

$$\max_{u^k_t \in \mathcal{U}^k_t} (r_t + \xi_{t+1})u^k_t$$

where k is the iteration of our algorithm introduced in the next section.

A vehicle flow feasibility constraint is needed to ensure that a vehicle flow $\mathbf{u}$ satisfies the conservation of vehicles into and out of each node N. The constraint is

$$\sum_{j|(i,j)\in A} u^k_{ijt} - \sum_{j|(j,i)\in A} u^k_{jit} = b^k_t(i) \quad \forall i \in N, \;\; k \in K \tag{28}$$

$$u^k_{ijt} \in \mathcal{B} \quad \forall (i, j) \in A, \;\; k \in K \tag{29}$$

where $b^k_t(i)$ is 1 if node i is a supply node in the network, 0 if node i is a transhipment node, and -1 if node i is a demand node. $\mathcal{B}$ is the set of binary numbers. The values for $b^k_t(i)$ are set as $R_{it} = b^k_t(i)$ for the first M equations representing the source flows, $b^k_t(i) = 0$ for the next M flows from the first level to the second level and the sink is represented by $b^k_t(M + M + 1) = -\sum_{i=1}^{M} R_{it}$.

Equivalently, a matrix N^k_t is defined as the network routing matrix requiring that if vehicle k enters a node, it must also leave that node. Each column $(N^k_t)_{ij}$ in the matrix corresponds to the variable u_{ijt}. The column of $(N^k_t)_{i,j}$

has a $+1$ in the ith row and a -1 in the jth row. Let $b_t^k = [b_t^k(1), b_t^k(2), \ldots, b_t^k(N)]$. The rest of the entries are zero. The feasible region $\mathcal{U}_t^k$ at time t is defined as:

$$N_t^k u_t^k = b_t^k \tag{30}$$

$$u_{ijt}^k \in \mathcal{B} \quad \forall (i, j) \in A \tag{31}$$

where N_t^k is referred to as the arc-incidence matrix.

Therefore the complete network problem for the dynamic recursion for a fixed algorithm iteration k is:

$$\max_{u_t^k} \quad (r_t^k + \xi_{t+1}^k)^T u_t^k \tag{32}$$

subject to

$$N_t^k u_t^k = b_t^k \tag{33}$$

$$u_{ijt}^k \in \mathcal{B} \qquad \forall (i, j) \in A \tag{34}$$

V. Uniform Travel Time ADP Algorithm

A realization of task arrivals is taken according to the Markov chain in the previous section. Dependent on the sample realization of the task arrivals at any stage

$$r_{ij}(t) = \begin{cases} V_j & \text{if task } j \text{ has arrived and has not yet been serviced} \\ 0 & \text{otherwise} \end{cases}$$

where V_j is the value of the arriving target.

The essence of Godfrey and Powell's approach[7] for this integer dynamic problem is that the stochastic optimization problem can be solved by forming the dynamic programming recursion in terms of the post-decision state:

$$J_t^u(S_t) = \max_{u_t \in \mathcal{U}_t} C_t(S_{t-1}^u, \omega_{t-1}, u_t)) + E_t \left\{ J_{t+1}^u(S_{t+1}^u) \right\} \tag{35}$$

where the expectation is taken over all possible outcomes of $\hat{S}_{t+1}$. Then by taking a single sample realization, the expectation is dropped to form

$$J_t(S_t) = \max_{u_t \in \mathcal{U}_t} C_t(S_{t-1}^u, \omega_{t-1}, u_t)) + \tilde{J}_t^u(S_t^u) \tag{36}$$

The Uniform Travel Time Adaptive Dynamic Programming Algorithm is:

STEP 0 Initialization: Initialize $\xi_t^0 = 0, t = 0, \ldots, T,$ Set $n = 0$

STEP 1 Do while $n \leq N$: Choose $W^n = (W_0^n, W_1^n, \ldots, W_{T-1}^n) \in \Omega$ according to current state of the Markov chain for each arriving task.

STEP 2 FORWARD PASS Do for $t = 0, 1, \ldots, T-1$:

2a Form the network problem of Equation to solve

$$\arg \max_{u_t^n \in U_t^n} (r_t^n + \xi_{t+1}^n)^T u_t^n$$

subject to its constraints to obtain u_t^n and the dual values π_{it}^n, for $i = 1, \ldots, M$

2b Update $R_{j,t+1}^n = u_{ijt}^n$ and $L_{t+1}^n = L_t^n + W_t^n - L_t^e$ where

$$L_{jt}^e = \begin{cases} 1 & \text{if } u_{ijt}^n \text{ and } L_{jt}^{n+} = 1 \\ 0 & \text{otherwise} \end{cases}$$

2c Letting

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

Update linear approximation. $\xi_{it}^{n+1} = (1-\alpha^n)\xi_{it}^n + \alpha^n \pi_{it}^n,$ for $i = 1, \ldots, M$ if vehicle not present at time t for node i.

If vehicle present at time t for node i update linear approximation by $\xi_{it}^{n+1} = (1-\alpha^n)\xi_{it}^n + \alpha^n(r_t^k + \xi_{t+1}^n),$ for $i = 1, \ldots, M$ since only one vehicle can be located at each node.

STEP 3 Return ξ^N

The distance constraints for the vehicles are satisfied since the feasible set is modified after solution of each subproblem in the forward pass to include the determination of the reachable set of each vehicle.

In step **0** we initialize our linear estimate vector ξ_t^0. In step **1** we begin out loop for N iterations and choose a new sample realization $W^n = (W_0^n, W_1^n, \ldots, W_{T-1}^n)$ for each iteration n. In step **2** for each time t we solve the dynamic programming recursion using the current estimate ξ_{t+1}^n for the value-to-go. We obtain the dual variables $\pi_{it}^n,$ for $i = 1, \ldots, M$ where M is the number of nodes in the network from the XPressMP solver to update our linear estimates. These dual variables are the shadow cost of having one more vehicle at each node at time t and are therefore used to update ξ_t^{n+1} in step **2c** that will be used in iteration $n+1$. The state dynamics are in step **2b** to construct the problem for the dynamic programming recursion in time $t+1$. In step **2c** we update ξ_{it}^{n+1} using the dual variables if a vehicle was not present at node i at time t or we update ξ_{it}^{n+1} with the actual contribution to the objective function if a vehicle was located at node i at time t. This updated linear vector ξ_t^{n+1} is used in the $n+1$ iteration of the algorithm. The algorithm returns to step **1** and repeats N times finally returning the linear matrix for the value-to-go approximation $\xi_t^N, t = 1, \ldots, T$. From this vector a decision for the next step of the stochastic problem can be made.

A. Value Function Approximation

Unlike in the fleet management problems addressed by Godfrey and Powell,[7] the UAV routing problem has few vehicles that are traversing the network and these vehicles are constrained by their maximum travel time in the system.

To simplify the value-to-go value function $\tilde{J}_t^{u,n}(S_t^u(u_t))$, we use a linear approximation

$$\tilde{J}_t^u(S_t^u) = \xi_t^T u_t \tag{37}$$

where ξ_t is an estimate of the slope of J_t^u with respect to the supply of UAVs. ξ_t is referred to as the vehicle gradient. u_t is the assignment decision vector

Using this linear value function approximation, our problem is

$$\max_{u_t \in \mathcal{U}_t} \sum_{i \in \{\mathcal{J}_t | R_{it} > 0\}} \sum_{j \in \{\mathcal{J}_{t+1}\}} (r_{ijt} + \xi_{j,t+1}^n) u_{ijt}$$

The question remains how to obtain these linear estimates. We address this next.

B. Determining Subgradients

For the linear estimate, we would like to find the change in the approximate value-to-go value function, $\tilde{J}_t^u(S_t^u)$. This can be done through looking at the integer problems subgradients as shown in this section.

Consider the following network formulation

$$F(u) = \min\ c^T u \tag{38}$$

subject to

$$\mathcal{N}u = b \tag{39}$$

$$u \leq ub \tag{40}$$

$$u \geq 0 \tag{41}$$

where c is a vector of arc values, u is a vector of link flows, $\mathcal{N}$ is a standard node-incidence matrix and b is the vector of node sources or sinks assume the convention

$$b_i = \text{flow out of node } i - \text{flow into node } i$$

The vector u is the upper bound on the arcs. Let π be the vector of duals for the constraints 40 commonly called node potentials in the literature and let v be the vector of duals for the constraints 41.

The dual π_i is a subgradient of $F(u^*)$ with respect to the right hand side of the constraints b and satisfies:

$$D_i^- \leq \pi_i \leq D_i^+ \tag{42}$$

where D_i^- and D_i^+ are the defined by considering the classical linear network problem 5.2-41.

Assume the linear network problem (5.2-41 has been solved. Consider the following perturbed problem:

$$F_i(u(\epsilon)) = \min_{u(\epsilon)} c^T u(\epsilon) \tag{43}$$

subject to

$$\mathcal{N}u(\epsilon) = b + \epsilon e_i \tag{44}$$

$$u(\epsilon) \leq u \tag{45}$$

$$u(\epsilon) \geq 0 \tag{46}$$

where e_i is a vector of zeros with a one for the row corresponding to node i.

Define directional derivatives D_i^+ and D_i^- as:

$$D_i^+ = \lim_{\epsilon \to 0^+} \frac{F_i(u^*(\epsilon)) - F(u^*)}{\epsilon} \tag{47}$$

and

$$D_i^- = \lim_{\epsilon \to 0^-} \frac{F_i(u^*(\epsilon)) - F(u^*)}{\epsilon} \tag{48}$$

The following result is shown in:[11]

Theorem 2. *The vector of duals π is a subgradient of $F(u^*)$ with respect to the right hand side vector b.*

Proof: Let

$$z(b) = \min_u c^T u = \max_{\{\pi, v\}} \pi^T b + v^T u \tag{49}$$

Let $\{\pi^*, v^*\}$ be the optimal duals for u^* and let $\bar{u}^*$ and $\{\bar{\pi}^*, \bar{v}^*\}$ be the optimal primal and dual solutions for the problem with right hand side $\bar{b}$. Then

$$z(b) - z(\bar{b}) =$$

$$= [\pi^{*Z} b + v^{*Z} u] - [\bar{\pi}^{*Z} \bar{b} + \bar{v}^{*Z} u] \tag{50}$$

$$\geq [\bar{\pi}^{*Z} b + \bar{v}^{*Z} u] - [\bar{\pi}^{*Z} \bar{b} + \bar{v}^{*Z} u] \tag{51}$$

$$\geq \bar{\pi}^{*Z} (b - \bar{b}) \tag{52}$$

where the first inequality follows since $\{\bar{\pi}^*, \bar{v}^*\}$ are not necessarily optimal for the problem with vector b in lieu of $\bar{b}$. ■

In this class of problems, it is easy for a current solution at iteration n, u^n, to find the duals, π, and calculate $g_t(u_t^n)$ where n denotes the nth iteration of our algorithm.

$$v_{it}^n = \begin{cases} g_t(u_t^n + \Delta_i, \omega_t^n) - g_t(u_{it}^n, \omega_t^n) & \text{if location } i \text{ has not yet been serviced} \\ g_t(u_t^n, \omega_t^n) - g_t(u_t^n - \Delta_i, \omega_t^n) & \text{if location } i \text{ has been serviced by } u_t^n \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

where Δ_i is the effect of adding one vehicle to location i.

However, it is very difficult to calculate the exact values of $\mathbb{E}g_t(u_t)$. Instead, using stochastic subgradient techniques, the fact that v_t^k satisfies

$$\mathbb{E}\left\{v^k | u^k\right\} \in \delta F(u^k) \tag{54}$$

where δF is the subdifferential of F is used to determine the stochastic subgradients. At current points u_t^n, $n = 0, 1, \ldots,$ random observations are taken, $(\omega_0, \ldots, \omega_{t-1})$ and data obtained is used to calculate realizations of these subgradients which correspond to exact subgradients only in the mean.

We are then interested in the dual values for the first M constraints on the set of equations $N_t u_t = b_t$ which translate to the marginal value or shadow price of having one more resource at a given task location. We define these dual values π_i for the ith task location. The network problem is solved using XpressMp, at iteration n and time t the dual values $\pi_{it}^n, i = 1, \ldots, M$ are calculated and the estimated marginal values, $\tilde{v}_t^n$ are updated. At each iteration n, a new set of flows, u^n, is realized. This creates a situation where estimates may vary considerably from one iteration to the next. For this reason the estimates are smoothed according to

$$\tilde{v}_{it}^{n+1} = (1 - \alpha^n)\tilde{v}_{it}^n + \alpha^n \pi_{it}^n \tag{55}$$

where $\alpha \in (0, 1)$ is our smoothing factor so that the current value is averaged with the previous n iterations.

One choice for our linear estimate is

$$\xi_t^n = \bar{v}_t^n$$

Through this procedure the approximate value-to-go value function is updated for each task location for each time step. Unlike approximate policy iteration where the value-to-go function is updated only for the current state, use of the dual information provides that the value-to-go function value of all locations is updated. In other words, approximate policy iteration updates occur as an OR operator. This OR operator means that the vehicle, from its current state, can either transition to one state or another until all states are explored from its current position. Instead, in our approach, we see what the effect would be at time t of a vehicle being at a location whether or not a vehicle can actually be there at time t. In other words, our approach updates occur as an AND operator answering the question of what one additional vehicle at each location would have on the objective function.

C. Convergence

For complicated problems such as the uniform travel time problem, convergence is much more difficult to prove than the simpler discrete two-stage news vendor whose convergence is proven by Powell and Topaloglu[13] under the assumption that all slopes are sampled with positive probability. As of this writing, formal proofs of convergence do not exist for this technique.

To obtain convergence the functional form of an approximation has to reasonably capture the true value function. To do this, the structure of the problem needs to be exploited so that at each Monte Carlo simulation, a visit to one state provides improved estimates of the value of visiting a large number of other states.

For future work, the proof of convergence most likely would follow a similar path of the research done on quasi-gradient techniques by Ermoliev.[6]

D. Model Predictive Control

The term model predictive control does not designate a specific control strategy, but rather a method of using a representative model, in this case the adaptive dynamic programming formulation, to obtain controls or decisions for a short time period into the future. The model is representative of the current state and must be resolved when

the state "changes significantly." In the case of UAV planning and control under uncertainty of arrivals, an arriving target could constitute a significant state change and require resolving the formulation. The real-time aspect of the problem dictates that these problems be solvable in minutes within the model predictive control framework.

E. Experimentation and Results

We will consider two networks given in Figs. 4 and 5. We begin with Fig. 4 and consider a deterministic example where vehicles are allowed two steps or moves. This example demonstrates how updating of the function value occurs. The targets are at their locations starting at time 0 until they are instantaneously served upon being colocated with a vehicle at which time the task leaves the system. The target values are given in Table 1. Each example starts with two vehicles at node 0 in their respective networks.

The algorithm is run for 100 Monte Carlo samples, which are trivial in this case since all targets arrive with probability 1, and an optimal objective value of 10021 is obtained. By inspection, this value is easily verified, but, more importantly, we demonstrate how the answer was obtained and the fact that enough samples must be taken so that the averaging converges.

The dual values of the first dynamic programming recursion are meaningless because ω_0 has no meaning in the value function approximation. Therefore, we begin by looking at the linear estimates derived from the dual values of the second dynamic programming recursion that provide the linear estimates of the value-to-go function, $\tilde{J}$, for the first recursion. The dual values are considered only where there is not a vehicle. For nodes that have vehicles we take the actual value that is realized in solving the dynamic programming recursion to update the linear estimate.

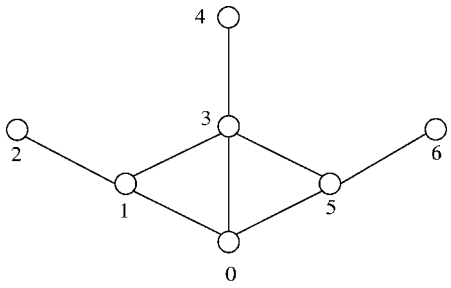

Fig. 4 Example of the one-step problem.

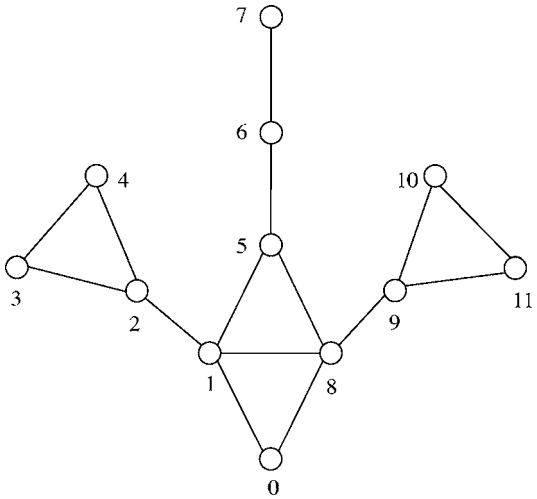

Fig. 5 Example 2 of the one-step problem.

Table 1 Target values for figure 4 example 1.

Node	Value
0	0
1	10
2	10
3	1
4	10000
5	10
6	10

Table 2 Dual values for figure 4 example 1.

$\xi_1^0 =$	{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0}
$\xi_1^1 =$	{0.0, 10.0, 0.0, 10000.0, 0.0, 10.0, 0.0}
$\xi_1^2 =$	{9.0, 10.0, 0.0, 10000.0, 0.0, 10.0, 9.0}

The linear estimates for the first three Monte Carlo samples are given in Table 2 where ξ_j^i represents the estimates after i Monte Carlo samples at the j dynamic programming recursion.

Interestingly, on the second Monte Carlo sample the optimal answer is obtained of 10021. At this point we could conjecture that for a deterministic example of n steps, n Monte Carlo samples are needed to obtain an optimal answer or $n-1$ updates to the linear estimates are needed before an optimal answer. The linear estimate used in the second Monte Carlo sample for the first dynamic programming recursion is ξ_1^1 and simply shows that if we send a vehicle to node 3 then we would get the value at node 3 plus 10000.0. The value for the linear estimate of the value-to-go for node 1 and node 5 is 10 from the actual realization of the assignment from the solution of the forward dynamic programming recursion using ξ_1^0. In the realization of this solution with objective value of 40 the two vehicles has one vehicle first traveling to node 1 and the other vehicle first traveling to node 5. Then the vehicle at node 1 travels to node 2 and the vehicle at node 5 travels to node 6. On the second Monte Carlo sample one vehicle travels the path from node 0 to node 1 to node 2 while the other vehicle travels from node 0 to node 3 to node 4 for the optimal objective value of 10021. On the following Monte Carlo samples this solution never changes for as the 1000 iterations experimentally run.

Next for Example 2 we look at 4 step deterministic example in Figure 5 with values given in Table 3. Again, all targets are at their locations starting at time 0 until they are instantaneously served upon being colocated with a vehicle at which time the task leaves the system. The example starts with two vehicles at node 0.

Table 3 Target values for figure 5 example 2.

Node	Value
0	0
1	200
2	200
3	200
4	200
5	0
6	0
7	10000
8	200
9	200
10	200
11	200

Table 4 Dual values for Fig. 5 example 2.

$\xi_1^1 =$	{0.0, 200.0, 200.0, 200.0, 200.0, 0.0, 10000.0, 0.0, 200.0, 200.0, 200.0, 200.0}
$\xi_2^1 =$	{0.0, 0.0, 200.0, 200.0, 200.0, 0.0, 10000.0, 0.0, 0.0, 200.0, 200.0, 200.0}
$\xi_3^1 =$	{0.0, 0.0, 200.0, 200.0, 0.0, 0.0, 10000.0, 0.0, 0.0, 200.0, 200.0, 0.0}
$\xi_1^2 =$	{0.0, 560.0, 380.0, 380.0, 380.0, 9000.0, 10000.0, 9000.0, 560.0, 380.0, 380.0, 380.0}
$\xi_2^2 =$	{0.0, 180.0, 560.0, 200.0, 380.0, 9000.0, 10000.0, 9000.0, 180.0, 560.0, 200.0, 380.0}
$\xi_3^2 =$	{0.0, 0.0, 200.0, 200.0, 0.0, 0.0, 10000.0, 0.0, 0.0, 200.0, 200.0, 0.0}
$\xi_1^3 =$	{145.8, 1175.6, 542.0, 542.0, 396.2, 9810.0, 17290.0, 9810.0, 14686.4, 542.0, 542.0, 396.2}
$\xi_2^3 =$	{0.0, 196.2, 592.4, 200.0, 396.2, 17910.0, 10000.0, 9810.0, 358.2, 430.4, 362.0, 396.2}
$\xi_3^3 =$	{0.0, 0.0, 200.0, 200.0, 0.0, 0.0, 10000.0, 0.0, 162.0, 200.0, 200.0, 162.0}

The algorithm is run for 100 Monte Carlo samples, which are trivial in this case since all targets arrive with probability 1, and an optimal objective value of 11000 is obtained. We note that the optimal solution is obtained after 4 Monte Carlo samples or 3 updates of the estimates and the solution does not change for the remaining 97 iterations.

We begin by looking at the linear estimates derived from the dual values of the second dynamic programming recursion that provide the linear estimates of the value-to-go function, $\tilde{J}$, for the first recursion and continue our analysis through the linear estimates of the third dynamic programming recursion. The linear estimates for the second through third time steps and the second through fourth Monte Carlo samples are given in Table 4 where ξ_j^i represents the estimates after i Monte Carlo samples at the j dynamic programming recursion.

The updates build upon each other so that the first three linear approximation vectors after the first Monte Carlo approximation ξ_j^1 are the one-step value-to-go values. The second three linear approximation vectors after the second Monte Carlo approximations ξ_j^2 are the two-step value-to-go values scaled by the averaging factor $\alpha = .9$. Thus the value-to-go functions build up to indicate the value-to-go from a given location at a given time dependent on the locations that were visited in earlier recursions since those task values will no longer be present.

In our next example we depart from the solely deterministic arrivals and have the arrival at node 7 follow a Markov chain with probability of arrival at each time period of p if the target has not yet arrived and 0 otherwise. We vary p to find the threshold for which the decision changes for a 4 time step problem with 2 vehicles starting at node 0. We make a decision for the first time step after 1000 Monte Carlo samples, then for the problem with 3 time steps remaining we run another 1000 Monte Carlo samples and make a decision. We continue until no time steps remain.

For values of $p < .02$ the solution is vehicle 1 traveling the sequence {0−1−2−3−4} and vehicle 2 traveling the sequence {0−8−9−11−10} for an objective value of 1600. The solution never changes since the optimal solution is the greedy solution. When $p > .02$ the solution becomes vehicle 1 traveling the sequence {0−1−5−6−7} and vehicle 2 traveling the sequence {0−8−9−11−10} for an average objective value that is greater than 1600 dependent on the value of p. For $p = .021$ the algorithm required 284.2 seconds and 298 iterations to converge to the optimal solution. For $p = .025$ the algorithm require 204.7 seconds and 212 iterations to converge to the optimal solution.

These results follow from the fact that we would expect the decision to change when the task at node 7 arrives significantly often in our Monte Carlo samples. This example demonstrates the effectiveness of this technique to stochastic arrivals of tasks. The critical decision in the sequence for this example is made after the first decision in the sequence since the sequence for vehicle 1 is always identical. Therefore only three time steps are remaining when the critical decision must be made. The decision would change when, on average, a value of 600 or more is present at node 7. If we calculate this value analytically for our Markov chain we set

$$1 - (1 - p)^3 = .06 \tag{56}$$

and solve to obtain $p = .0204139$. Our simulation results therefore is consistent with the analytical solution.

Since convergence is not proven we conduct this last experiment again but with the initial estimates set to an upper bound of 20,000, greater than the sum of all target arrivals. We find that once again the algorithm behaves correctly and our simulation remains consistent with the analytical solution. Finally, we set the initial estimates randomly

between 0 and 100,000 and run the experiment again for 100 differing starting estimates and find that our simulation results remain consistent with the analytical solution.

VI. Conclusion

Approaching the UAV routing problem using Adaptive Dynamic Programming offers a tractable framework in which to solve these difficult problems. A post-dynamic programming formulation offers a framework in which, after only a few iterations, the functional approximations for the cost-to-go functions represent a value closer to reality. While greater solution times will be needed, the use of more complex functional estimates and larger networks appear to offer solutions that better account for uncertainty.

Acknowledgments

I am grateful for the guidance of Professor David Castañón of Boston University and Dr. Stephan Kolitz of the Charles Stark Draper Laboratory for their guidance and encouragement in developing this work as part of my graduate studies.

References

[1]Richard E. Bellman, *Dynamic Programming*, Princeton University Press, 1957.

[2]Dimitris Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 1995.

[3]Dimitris Bertsekas, Benjamin Van Roy, Yuchun Lee, and John Tsitsiklis, "A Neuro-Dynamic Programming Approach to Retailer Inventory Management," In *Proceedings of the 36th IEEE Conference on Decision and Control*, Vol. 4, pp. 4052–4057, 1997.

[4]Dimitris Bertsekas and John Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5]William B. Carlton, *A Tabu Search to the General Vehicle Routing Problem*. A PhD dissertation, University of Texas, Austin TX, 1995.

[6]Gambardella, L.M., Taillard, E. and Agazzi, G. *Numerical Techniques for Stochastic Optimization*, chapter Stochastic Quasigradient Techniques, Springer-Verlang, 1988.

[7]Gregory Godfrey and Warren Powell. "An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management, I: Single Period Travel Times," *Transportation Science*, Vol. 36, pp. 21–39, 2002.

[8]Gregory Godfrey, Warren Powell, Tassio Carvalho, and Hugo Simao. "Dynamic Fleet Management as a Logistics Queueing Network," *Annals of Operations research*, Vol. 61, pp. 165–188, 1995.

[9]Katerina P. Papadaki and Warren B. Powell, "A Monotone Adaptive Dynamic Programming Algorithm for a Stochastic Batch Service Problem," *European journal of Operations Research*, Vol. 142, pp. 108–127, 2002.

[10]Katerina P. Papadaki and Warren B. Powell, "An Adaptive Dynamic Programming Algorithm for a Stochastic Multiproduct Batch Dispatch Problem," *Naval Research Logistics*, Vol. 50, pp. 742–769, 2003.

[11]Warren B. Powell, "A Review of Sensitivity Results for Linear Networks and a New Approximation to Reduce the Effects of Degeneracy," *Transportation Science*, Vol. 23, pp. 231–243, 1989.

[12]Warren B. Powell and Tassio A. Carvalho, "Dynamic Control of Logistics Queueing Networks for Large-scale Feel Management," *Transportation Science*, Vol. 32, pp. 161–175, 1998.

[13]Warren B. Powell and Huseyin Topaloglu, "An Algorithm for Approximating Piecewise Linear Concave Functions from Sample Gradients," *Operations Research Letters*, Vol. 31, pp. 67–76, 2003.

[14]Marius M. Sisson, "Applying Tabu Search to Wind Influenced, Minimum Risk, and Maximum Expected Coverage Risk." *M.S. thesis, Air Force Institute of Technology, Wright-Patterson AFB OH, 1997.*

[15]Warren B. Powell and Michael Z. Spivey, "The Dynamic Assignment Problem," *Transportation Science*, Vol. 38, pp. 399–419, 2004.

[16]Martin L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 1994.

James Neidhoefer
Associate Editor