Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds

WEAPON TRADEOFF ANALYSIS USING DYNAMIC PROGRAMMING FOR A DYNAMIC WEAPON TARGET ASSIGNMENT PROBLEM WITHIN A SIMULATION

Darryl Ahner
Carl Parson

Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH 45433-7765, USA

ABSTRACT

We consider the sequential allocation of differing weapons to a collection of adversarial targets with the goal of surviving to destroy a critical target within a combat simulation. The platform which carries the weapons proceeds through a set of sequential stages and at each stage potentially engages targets with available weapons. The decision space at each stage is affected by previous decisions and the probability of platform destruction. Simulation and dynamic programming are then used within a larger dynamic programming framework to determine allocation strategies and develop value functions for these mission sets to be used in future, larger and more complex simulations. A simple dynamic programming example of the problem is considered and used to generate a functional approximation for a more complex system. The developed methodology provides a tractable approach to addressing complex sequential allocation of resources within a risky environment.

1 INTRODUCTION

The subject of this paper is the sequential allocation of two competing weapons to a collection of adversarial targets that affect the overall effectiveness of an adversarial system of systems for use within a combat simulation. It is a subject that has become increasingly important with the development of more complex weapon systems. For example, the effectiveness of two differing missiles yield varying effects on targets and the optimal strategy could consist of a mix of these two missiles which may differ by range, probability of kill, and number able to be carried. To compare these two weapon types, the platform which carries the weapons must proceed through a predefined set of sequential stages and at each stage a decision must be made to engage targets for that stage and, if so, which missile to use.

The decision space at each stage is affected by previous stage decisions and the probability of platform destruction at each stage is a function of the current state of the adversarial system of systems. Therefore, while it is generally less risky for the platform to ignore the current target and continue its mission, the cumulative risk to mission completion may result in the optimal solution at that stage being to engage the target. Additionally, while one missile may have a higher probability of destroying the target, future stages may dictate the use of the less capable missile.

The special structure of the problem is exploited to recursively update functional approximations representing future decisions using the problem's subgradient information. A dynamic programming methodology is developed to determine allocation strategies and develop value functions for these mission sets for use in future, larger and more complex simulations. This value function must take into account full state information to include missiles of each type remaining, the current state of the adversarial system of systems, and remaining stages to accomplish the destruction of the primary target.

First posed by Manne (1958), the weapon target assignment problem has provably optimal solutions when all the weapons are identical (denBroder et al. [1958] and Katter [1986]) and when the targets can receive at most one weapon (Chang et al. [1987] and Orlin [1987]). Previous work addressed performing tasks in uncertain environments where vehicles may fail (Castanon et al. [2008]). A more comprehensive survey of the weapon-target assignment problem can be found in (Cai et al. [2006]) This paper presents a strategy based on these algorithms to determine valid subgradient information.

2 DYNAMIC PROGRAMMING USING A POST DECISION STATE

Godfrey and Powell (2002) introduce an adaptive dynamic programming algorithm for stochastic dynamic resource allocation problems. Consider the general problem where the new task arrival information, W_T, is independent of the decision vector x_t and the state $S_{t+1} = f_1(S_t, x_t, W_t)$ where f_1 is a function describing the system dynamics.

Let $C_t(S_t, x_t)$ be the contribution received in period t given the state S_t and decision $x_t \in X$. The information that arrives in time t is W_t, a random variable generated with a known probability distribution. It is well known that problems that maximize these costs over a finite time horizon can be solved by Bellman's (1957) optimality equations:

$$J_t(S_t) = \max_{x_t} \mathrm{E}_t \left\{ C_t(S_t, x_t) + J_{t+1}(S_{t+1}(S_t, x_t, W_t) \mid S_t \right\} \tag{1}$$

$J_t(S_t)$ is commonly referred to as the value-to-go. Equation 1 requires us to take into account the impact of current decisions on future decisions which depend on future policies. As shown in Puterman (1994) we write the expectation for the discrete case as

$$J_t(S_t) = \max_{x_t} C_t(S_t, x_t) + \sum_{s \in S_{t+1}} p(s' \mid S_t, x_t) J_{t+1}(s') \tag{2}$$

where S is the set of potential states and $p(s' \mid S_t, x_t) =$ the probability the system will be in state s' given that the current state is S_t and we take action x_t.

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

As an alternative, consider a post-decision state S_t^x at time t. The post decision state evolves according to a state equation which has the form $S_t^x = f_2(S_t, x_t)$ where f_2 is a function describing the post-decision state dynamics. S_t^x always exists by simply defining $S_t^x = (S_t, x_t)$. This post-decision state is the state that is acted upon by a decision x_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, x_t.

In Powell (2007), noting that S_{t+1} is a function of S_t^x 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:

$$J_t^x(S_t^x) = \mathrm{E} \left\{ \max_{x_{t+1}} C_{t+1}(S_t^x, W_t, x_{t+1}) \quad J_{t+1}^x(S_{t+1}^x) \mid S_t^x \right\} \tag{3}$$

where the value function is indexed with the superscript x 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^x, the information W_t is needed in order to determine x_{t+1}.

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

$$x_{t+1} = \arg\max_{x_{t+1}} C_{t+1}(S_t^x, W_t, x_{t+1}) + J_{t+1}^x(S_{t+1}^x(S_{t+1}, x_{t+1})) \quad (4)$$

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 $J_t^x(S_t^x)$ for each post-decision state which is generally not feasible for combinatorial problems. In the next section we discuss how to mitigate this issue.

3 THE GENERAL ADAPTIVE DYNAMIC PROGRAMMING ALGORITHM

Godfrey and Powell (2002) introduce an adaptive dynamic programming algorithm for stochastic dynamic resource allocation problems. Again, consider the general problem as defined where the new task arrival information, W_t, is independent of the decision vector x_t and the state S_t. Assume we have an approximation of the value-to-go function $J_t^{n-1}(S_t^x)$ where n is the current iteration. Given a realization $\left\{\widetilde{W}_0, \widetilde{W}_1, \ldots \widetilde{W}_{T-1}\right\}$, we compute x_0 according to

$$x_0 = \arg\max_{x_0 \in X_0} C_0(S_0, x_0) + J_1^{n-1}(S_0^x(x_0)) \quad (5)$$

We use a forward algorithm recursively from time $t = 1 \ldots . T-1$ to compute $S_1, x_{1,\ldots}$ according to

$$x_t = \arg\max_{x_t \in \aleph_t} C_t(S_{t-1}^x, \widetilde{W}_{t-1}, x_t) + \widetilde{J}_{t+1}^{x,n-1}(S_t^x(x_t)) \quad (6)$$

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

We 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 $\widetilde{J}_t^{n-1}(S_t^x)$ is updated which is problem dependent.

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

Step 1 Do while $n \le N$: 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

$$x_t = \arg\max_{x_t \in \aleph_t} C_t(S_{t-1}^x, \widetilde{W}_{t-1}, x_t) + \widetilde{J}_{t+1}^{x,n-1}(S_t^x(x_t))$$

2b Update the state S_t

2c Update the value function approximation $\widetilde{J}_t^{n-1}(S_t^x)$ using information collected from the optimization in step 2a

Step 3 Return J^N and use it to select x_0

This algorithmic approach is similar to approximate dynamic programming as shown in Bertsekas and Tsitsiklis (1996). 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. This proposed approach is explained further in Section 6.

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, and the methods used to update the value-to-go approximation. In the next sections we begin by exploring a simple dynamic programming example of the problem and then conjecture a functional approximation for future exploration.

4 SIMPLE DYNAMIC PROGRAMMING EXAMPLE FOR WEAPON TRADEOFF ANALYSIS

As an initial formulation, a finite space, infinite horizon dynamic programming problem is considered. This example consists of a platform with two weapon types available, each with a single use, facing two targets: a defensive target and a C^2 building. Each possible state is investigated to determine the optimal control based upon the previous state, decision and outcome. This formulation also includes the inherent risk of not destroying the defensive target prior to engaging the C^2 building, meaning that there is a nonnegative probability that any remaining weapons will be destroyed if the defensive target is functional.

State Space

Here our state consists of a quadruple with each index being a dichotomous variable denoting whether the element is functional (1), or destroyed (0). For the most simple example our quadruple will take on the form:

(Kinetic Energy (KE) Weapon, Directed Energy (DE) Weapon, Defensive Target, C^2 Target)

As an example, the state (1,0,1,1) means that weapon type 1 is still available and both targets are currently functional. For this formulation, our initial state is always (1,1,1,1).

Decision Space

At each stage we need to determine whether or not we should engage any remaining targets. What will drive this decision is the risk that our aircraft will get shot down in a future stage, which is a function of the state of the defensive target. Our decision will be in the form of a bivariate vector where the indices denote which target is to be engaged at the current stage. Each active weapon will be able to make one of three choices:

0 = Engage no target
1 = Engage the defensive target
2 = Engage the C^2 target

For example, the decision (1,2) would mean that the first weapon be shot at the defensive target and the second weapon be shot at the C^2 target. Given the initial state of our formulation, 9 decisions (or controls) can be applied, though this changes as the states transition.

Reward Function

Our example allows for us to obtain a reward for each target destroyed. We assume that the defensive target value, V_1, will be less than the value of the C^2 building as it is the ultimate goal. The risk is modeled in this example by including a nonzero probability of our aircraft being shot down. This probability is dynamic, and evolves as the adversarial treats change. Thus, our expected reward depends on the likelihood of being shot down prior to destroying the C^2 building.

Transition Function

Even for this simple of an example, several possible transitions are possible at each stage. These transitions are predicated on predetermined knowledge of the probability of killing a target. For our formulation we will assume

$p_i^{def} \stackrel{\text{def}}{=}$ the probability of killing the defensive target with weapon type i (7)

$p_i^{C^2|def} \stackrel{\text{def}}{=}$ the probability of killing the C^2 target with weapon type i given the defensive target is functional (8)

$p_i^{C^2|no\ def} \stackrel{\text{def}}{=}$ the probability of killing the C^2 target with weapon type i given the defensive target is destroyed (9)

where i indexes the KE and DE weapon.

In addition, at each stage, the state can transition as a function of the risk of being shot down given the defensive target is still operational.

Terminal State

The terminal state happens when the aircraft has either been destroyed, has used all available weapons, or has destroyed the C^2 target.

5 SIMPLE EXAMPLE RESULTS

To obtain exact solutions for this problem formulation, value iteration is used. Value iteration is well suited for this problem because for each possible state, it takes into account the expected reward from making all possible decisions as well as the expected future reward. Therefore each state transition accounts for future information when making the current decision. Once this is done, the control which maximizes this function is selected and the algorithm is updated. Table 1 presents the three initial scenarios investigated for our formulation and Table 2 presents the optimal controls which are consistent with the format discussed in Section 4 above.

Table 1. Single shot probability of kill for weapon target combinations

	KE vs Def	KE vs C2|def	KE vs C2|no Def	DE vs Def	DE vs C2|def	DE vs C2|no Def
Scenario 1	0.6	0.1	0.95	0.6	0.1	0.95
Scenario 2	0.5	0.5	0.5	0.5	0.5	0.5
Scenario 3	0.6	0.3	0.8	0.8	0.1	0.95

Table 2. Optimal controls for feasible states

	(1,1,1,1)	(0,1,1,1)	(0,1,0,1)	(1,0,0,1)	(1,0,1,1)
Scenario 1	(1,0)	(0,1)	(0,2)	(2,0)	(1,0)
Scenario 2	(2,2)	(0,2)	(0,2)	(2,0)	(2,0)
Scenario 3	(2,1)	(0,1)	(0,2)	(2,0)	(0,0)

Given a homogeneous weapon set (Scenario 1), and subsequently changing to nonhomogeneity (Scenarios 2 & 3), our optimal controls provide substantial insight. For all scenarios, the defensive target value used was 500 and the C^2 target was valued at 1000. These values are arbitrary, so the results would be consistent so long as the higher valued target was double the value of the secondary target. Looking at state (1,0,1,1) for the first scenario, it is notable that the optimal control is to shoot the remaining (KE) weapon at the defensive target in lieu of targeting the C^2 target. This is a function of the greatly reduced marginal probability of kill as well as the risk of being shot down. Simply put, the expected reward of shooting the defensive target is greater than that of shooting the C^2 target. For future formulations, weighting the primary target by a large enough margin may be an area of added complexity for investigation, to include adding it as a factor in an experimental design.

The results presented in Table 2 also demonstrate how the controls change when weapons capabilities have shifted. When the single shot p_{kill}s are identical for each weapon-target pair (Scenario 2), intuitively you would want to shoot your weapons at the highest valued target. Our method captures this. Conversely, in Scenario 3 the DE weapon is more effective against the defensive target, and the KE weapon, has a greater chance of killing the C^2 target if the defensive target is functional. As such, our results exhibit this relationship with the presented controls.

As a means for further investigation, a 12 run latin-hypercube design was employed to gain additional insights into the behavior of the system. The design matrix is presented in Table 3, with the results shown in Table 4.

Table 3. Latin-hypercube design matrix

	KE vs Def	KE vs C2|def	KE vs C2|no Def	DE vs Def	DE vs C2|def	DE vs C2|no Def
Run 1	0.53332	0.1	0.1	0.51432	0.1	0.9
Run 2	0.9	0.1	0.35383	0.1	0.9	0.15127
Run 3	0.1	0.61813	0.11092	0.9	0.42802	0.1
Run 4	0.1	0.1	0.899	0.9	0.9	0.63508
Run 5	0.10806	0.9	0.83686	0.8935	0.1	0.9
Run 6	0.9	0.81229	0.1	0.9	0.84976	0.9
Run 7	0.82494	0.9	0.1	0.1	0.1	0.1
Run 8	0.9	0.81229	0.1	0.9	0.84976	0.9
Run 9	0.1	0.65489	0.27806	0.1	0.9	0.78298
Run 10	0.72835	0.9	0.9	0.7183	0.9	0.1
Run 11	0.9	0.1	0.84973	0.9	0.1	0.18768
Run 12	0.1	0.42209	0.9	0.1	0.12249	0.10017

Table 4. Optimal controls for latin-hypercube design points

	(1,1,1,1)	(0,1,1,1)	(0,1,0,1)	(1,0,0,1)	(1,0,1,1)
Run 1	(1,0)	(0,1)	(0,2)	(2,0)	(1,0)
Run 2	(1,2)	(0,2)	(0,2)	(2,0)	(0,0)
Run 3	(2,1)	(0,1)	(0,2)	(2,0)	(2,0)
Run 4	(0,1)	(0,2)	(0,2)	(2,0)	(0,0)
Run 5	(2,1)	(0,1)	(0,2)	(2,0)	(2,0)
Run 6	(1,2)	(0,2)	(0,2)	(2,0)	(2,0)
Run 7	(2,1)	(0,2)	(0,2)	(2,0)	(2,0)
Run 8	(1,2)	(0,2)	(0,2)	(2,0)	(2,0)
Run 9	(2,2)	(0,2)	(0,2)	(2,0)	(0,0)
Run 10	(1,2)	(0,2)	(0,2)	(2,0)	(2,0)
Run 11	(0,1)	(0,1)	(0,2)	(2,0)	(1,0)
Run 12	(2,2)	(0,2)	(0,2)	(2,0)	(2,0)

One trend is that if the probability of killing the defensive target is low (ex. 0.1), the optimal control is to shoot the weapon(s) at the C^2 target. Though intuitive, the risk incurred by not destroying the defensive target does not outweigh the expected value of destroying the C^2 target. Additionally, as the probability of killing the defensive target increases, destroying it becomes the dominant control. This is consistent with the exception of when the probability of killing the C^2 target is (practically) independent of the defensive target, as seen in run 6. Further validation of our methodology comes from runs 4 and 11. Because the single-shot probability of destroying the C^2 target given the defensive target is functional, the technique determines that it is better to save a weapon (the KE weapon in our case) for a later stage because of the likelihood that the other weapon will destroy the defensive target first. Finally, we discuss the difficulties that arise as the state and decision spaces occur, and present our method for future analysis.

6 USE WITHIN COMBAT SIMULATIONS AND FUNCTIONAL APPROXIMATION

6.1 Value Function Determination

As the size of the state and decision spaces increase, the complexity of the transition function does not lend itself to an exact solution. The approximate dynamic programming algorithm described previously uses the simple example to determine $\widetilde{J}_t^{n-1}(S_t^x)$, the approximate value-to-go function within a stochastic combat simulation. To accomplish this a predetermined path for the weapon platform is determined based on its current mission. The path is divided into uniform sequential decision stages each requiring time Δt. For a given time t, the current state, S_t^x, is defined by the platform location, number of weapons available and the number and location of defensive targets that can act on the platform in the next *m* stages. At each stage, a decision consists of whether to assign an available weapon to a single target or not, and which weapon to assign. The decision is then implemented within the simulation which continues for Δt time until another decision is required. The simulation performs the transitions to determine the state at the next decision epoch. The state space transitions are a function of the previous decision and the probabilities. This decision process continues within the combat simulation until either all weapon platforms are destroyed or the critical target is destroyed.

The decision process uses the previous simple example in an approximate dynamic programming framework as demonstrated by the following steps:

1) Consider only the next *m* stages
2) Use Monte Carlo methods to determine updated the conditional probabilities used in the simple example
3) Formulate and solve simple problem for all possible states S_t^x
4) Update $\widetilde{J}_{t+1}^{n-1}(S_t^x)$ for all possible states S_t^x
5) $x_t = \arg\max_{x_t \in \aleph_t} C_t(S_{t-1}^x, \widetilde{W}_{t-1}, x_t) + \widetilde{J}_{t+1}^{x,n-1}(S_t^x(x_t))$
6) Continue *k* times according to Sections 2 and 3 updating $\widetilde{J}_t^{n-1}(S_t^x)$

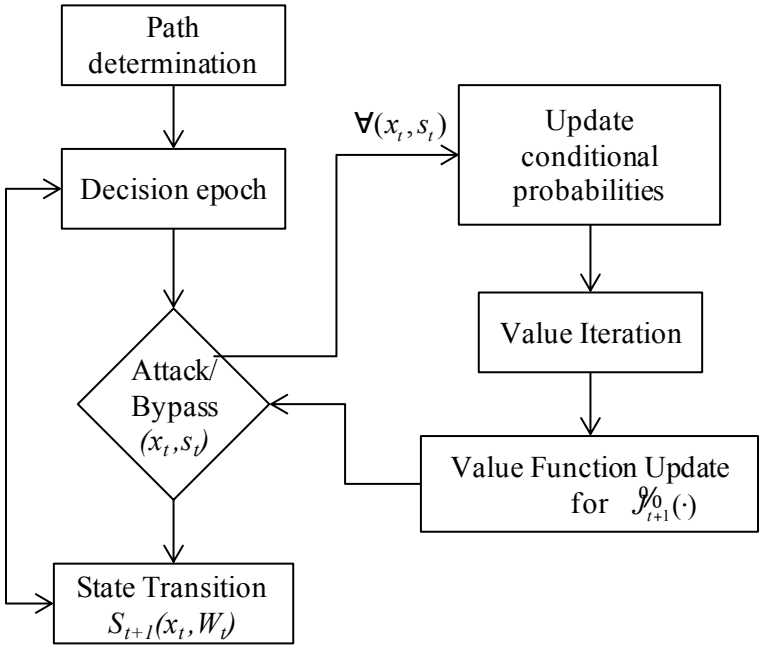

Figure 1. Value Function Determination framework

The value iteration presented earlier is used to determine the decision policy for the state at each decision epoch through the calculation of $\mathcal{V}_t(S_t^x)$. Since only the single-shot probability of destruction is known for each weapon-target pair, Monte Carlo experiments are conducted at the current state resulting in estimates for the conditional probabilities used in place of Equations 7-9. Just as in the simple example previously presented, intermediate defensive targets are valued less than the final critical target. The end result, after sufficient iterations, is a value function that represents the value-to-go approximation, $\mathcal{V}_t(S_t^x)$, for a given decision, x_t, for a given time t.

6.2 Simulation Implementation

Implementation within the simulation makes use of the determined value function to determine at each stage whether or not to engage targets with available weapons according to Figure 2.

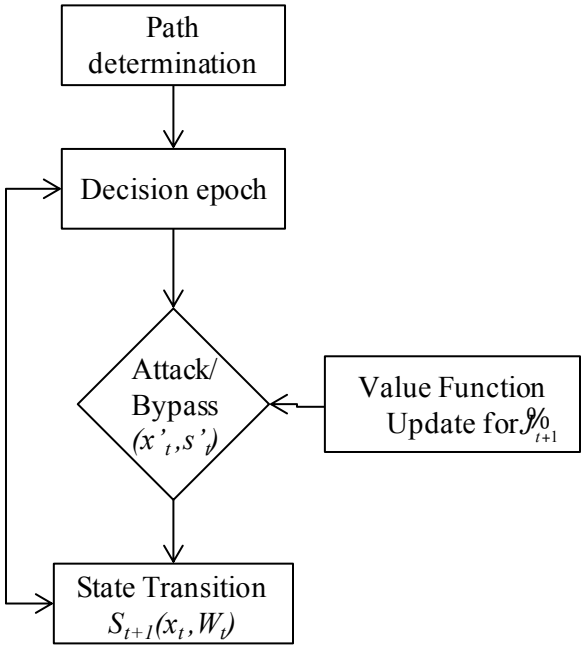

Figure 2. Combat Simulation Implementation

At each decision epoch, a decision is made based that maximizes the current one step expected reward plus the approximated value-to-go.

7 CONCLUSIONS

A solution methodology for the sequential allocation of differing weapons to a collection of adversarial targets with the goal to destroy a critical target within a combat simulation is presented. This approach demonstrates exploiting the special structure of the problem to recursively update functional approximations representing future decisions using the problem's approximate subgradient information. Simulation and dynamic programming are used within a larger dynamic programming framework to determine allocation strategies and develop a value function for these mission sets to be used in future, larger and more complex simulations. The developed methodology provides a tractable approach to addressing complex sequential allocation of resources within a risky environment.

REFERENCES

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

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

Castanon D., D Ahner. 2008. *Team Task Allocation and Routing in Risky Environments under Human Guidance*, Proceedings of the 47th IEEE Conference on Decision and Control, 1139-1144.

denBroeder. G., R.E. Ellison, L. Emerling, 1959. *On Optimum Target Assignments*, Operations Research, 7, 322-326.

Godfrey G., W. Powell, 2002. An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management, I: Single Period Travel Times, Transportation Science, 36, 21-39.

H. Cai, J. Liu, Y. Chen and H. Wang. 2006. Survey of the research on dynamic weapon-target assignment problem. *Journal of Systems Engineering and Electronics,* Vol 17, No.3, pp. 559-565.

Katter, J.D. 1986. A solution of the multi-weapon, multi-target assignment problem. Working Paper 26957, MITRE, McLean, VA.

Manne, A., 1958. *A Target-Assignment Problem*, Operations Research, 6, 346-351.
Powell W., 2007. Approximate Dynamic Programming, John Wiley & Sons, Hoboken, NJ.
Puterman, M., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley Publishing, New York.

AUTHOR BIOGRAPHIES

DARRYL K. AHNER is the Director, Center for Operational Analysis and an Assistant Professor at the Air Force Institute of Technology. He earned a M.S. in Applied Mathematics and a M.S. in Operations Research & Statistics from Rensselaer Polytechnic Institute in Troy, NY and a Ph.D. in Systems Engineering (Operations Research) from Boston University. His research interests include dynamic programming and using the combination of optimization and simulation for complex adaptive systems. His email address is <darryl.ahner@afit.edu>.

CARL PARSON is currently a PhD Student at the Air Force Institute of Technology. He earned a M.S. in Operations Research from the Air Force Institute of Technology. His research interests include stochastic optimization, simulation and optimization, dynamic programming and resource allocation. His email address is carl.parson@afit.edu>.