ABSTRACT

Many decision-and-control algorithms have been proposed for autonomous unmanned aerial vehicles (UAVs). The nature of this problem, with large decision spaces and the desire for optimal performance criteria, indicates that closed-form analysis of any approach is nearly impossible, suggesting a simulation-based performance evaluation of relevant scenarios. However, while effective simulation practices have been developed in the operations-research community, awareness of them in the decision-and-control community may be lower than desired for routine application. If applied correctly, these simulation techniques could have a major impact on the quality and effectiveness of UAV algorithms. This paper provides a concrete example that demonstrates that the marriage of UAV decision-and-control algorithms with proper simulation techniques can be done effectively and with great benefits for the UAV algorithm researcher, both in terms of the validity and quality of simulation results. The example used in this case is a study conducted for a stochastic UAV algorithm design. In particular, the study is to find the ''optimal'' sensitivity of a ''future-gain'' factor that attempts to balance future and present gains in a stochastic-approximate dynamic-programming solution to the problem faced by a team of UAVs searching in an uncertain environment for targets. This work can serve as a template for similar simulation experiments in this area. Experimental motivation and demonstration of the results are given.

INTRODUCTION

Research on control of unmanned vehicle systems has been increasing in recent years, especially in the area of outer-loop control of such vehicles, i.e. not the flight-level dynamics of such craft, but the decision-making capabilities that would enable these vehicles to complete complex tasks autonomously. See, for example, [O'Rourke et al. (2001)], [Hespanha and Kizilocak (2002)], [Jin et al. (2003)], [Richards et al. (2002)], [Castanon and Wu (2003)], [Cassandras and Li (2002)] and references therein. As can be seen in these examples, there are many complex factors involved in creating planning algorithms for this type of decision and control. Elements such as obstacles, threats, the presence of multiple vehicles of both friendly or hostile intent, uncertainty in the environment (i.e. unknown target locations, threat levels, etc.), and multiple tasks per vehicle can all contribute to a very complex problem. This is also true in the prior work of the authors of this paper ([Flint and Fernandez (2005)], [Flint et al. (2004a)], [Flint et al. (2003a)], [Flint et al. (2003b)], and [Flint et al. (2002)]).

In all of these papers, the main focus of the work is on the decision-and-control formulations of the vehicle-control problem. Given that actually fielding UAVs to validate the theoretical contributions is often prohibitively expensive and sometimes not possible, all of the papers include some simulation results instead. However, little attention is paid to the simulations that actually produce the numbers that appear in the papers. This is not a fault of the papers or their authors, since they are written for a target audience that is not interested in those details. But, this can serve to gloss over the important simulation aspects in the production of these algorithms.

This paper's contribution is to focus on those aspects, and provide a guide (or a reminder, to those already familiar with the topic) to UAV researchers about effective simulation. An example problem is drawn from the authors' cooperative-control research, and the practices of applying a simulation study in support of the research is given. While the cooperative-control problem is one that may be of interest if it were by itself, the use of the simulation techniques in support of the solution to the problem is the real focus of this paper. To this effect, several good practices have been highlighted throughout the paper. In particular, two specific simulation tools (defined in [Law and Kelton (2000)]) are applied to the current problem. The first is a statistically sound random-number generator (RNG). The second is a variance-reduction technique called *common random numbers* (CRN). By using these simulation tools, statistically valid and precise conclusions can be reached efficiently.

A motivating example is shown in Table 1, which gives the results for five different replications (chosen randomly) from

Simulation Analysis for UAV Search Algorithm Design Using Approximate Dynamic Programming

Dr. Matthew Flint

BAE Systems
matt.flint@baesystems.com

Dr. Emmanuel Fernandez

University of Cincinnati
emmanuel.fernandez@uc.edu

Dr. W. David Kelton

University of Cincinnati
david.kelton@uc.edu

APPLICATION AREAS: Unmanned Systems, Battle Management/ Command and Control and Modeling, Simulation and Wargaming

OR METHODOLOGIES: Dynamic Programming and Simulation

Table 1. Sample results from five replications. Each value is the number of targets (out of 10) that have been discovered by the search team at or before the given simulation time (in seconds). Note the wide variance in results between individual replications.

Simulation Time	300 s	600 s	900 s	1200 s	1500 s	1800 s	2100 s	2400 s	2700 s	3000 s
Rep 1	0	0	1	4	4	5	5	6	8	9
Rep 2	0	1	3	4	5	5	7	8	9	10
Rep 3	0	0	1	2	2	3	4	4	5	5
Rep 4	1	2	2	4	5	5	5	5	5	6
Rep 5	2	3	5	7	8	8	8	8	9	10

one (single) simulation trial presented later in the paper. Nothing differs between the replications but the randomness of the environment within the simulation; the algorithm and the parameters of the algorithm are not changing. Because of the wide variance from replication to replication, it can be very difficult within an experiment to gauge whether any change in performance of the search mission is due to the factors under test, or whether it is just due to random noise. The methods shown in this paper can then be used as a template for similar experiments in the area of cooperative control or beyond.

The problem demonstrated is the computation of the sensitivity to a *future planning weight factor,* which balances future and present gain in an approximate dynamic-programming optimization. But, more importantly, this paper presents and demonstrates simulation techniques that make this experimental comparison both efficient and accurate. The work presented here identifies key aspects, methods, and practices for a sound and reliable simulation study design.

The rest of this paper is organized as follows: In the summary section, the model and solution methodology used to create the path plans for multiple cooperating UAVs engaged in the search of an uncertain and dynamic environment is reviewed. Then, in the next section, the random-number generator and how it is utilized in the simulator and in the simulation study is discussed. In the following section, the common-random-numbers scheme is discussed and applied. The pilot simulation section gives a small demonstration that tests whether the techniques are effective in this particular instance. Then, in the next section, results of experiments using simulation are presented. These demonstrate how effective the techniques can be. The last section gives conclusions and a bibliography.

MODEL AND SOLUTION METHODOLOGY

The model and algorithmic solutions under consideration in this paper control the path-planning decision processes of multiple cooperating autonomous UAVs engaged in a search of an uncertain and risky environment. These solutions consist of methods to incorporate *a priori* and dynamic information about the search environment into a computationally feasible dynamic-programming (DP) solution methodology [Bertsekas (2000)]. In implementation, this solution methodology has been isolated from the simulator that is used to test it. The simulator and the solution algorithm have been coded in a custom, object-oriented C++ implementation. The simulator is of a fixed-increment-time-advance type, which means that the simulator advances simulation time for fixed periods of time before providing the advancing state of the world to the solution algorithm. In this case, this constructive simulation artifact generalizes well to the real-time case for this study because the algorithm under study is discretized in time such that, even though real vehicles travel in continuous time and space, they can make decisions only at discrete points in time (which are termed *decision time steps*).

The state of the environment, after discretization, is used by the DP algorithm to generate a path plan. The state of the environment is

represented internal to the solution algorithm by: (1) an information base (in the form of cognitive maps); and (2) the locations and headings of the vehicles. Let the ideal state (i.e. the one drawn from the best possible information available to all vehicles–not necessarily the truth state inside the simulator) of the environment be updated by every vehicle at every time step, and be denoted by x_k^*. The state perceived by an individual vehicle is denoted by x_k. Under ideal conditions, every vehicle will perceive the state as being the ideal state $(x_k^* = x_k,)$ but in the non-ideal case (which may arise because of, e.g., a limitation on communication bandwidth) each vehicle may have a different perception based on the information available to it. (This has been well documented in previous work, and so will not be covered in great detail in this paper. Please see [Flint and Fernandez (2005)], [Flint et al. (2003b)], and [Flint et al. (2004b)].)

Let J_k be what is called the ''cost to go,'' (which is accepted terminology, even though the formulation is in this case based on gain rather than cost). This represents the gain to be had from making decisions, and traveling the resulting paths, from time step k to some terminal (or end-of-mission) time step N for a planning vehicle. Then, the best path at time step k can be found from taking the arguments, at each time step, from expanding the dynamic-programming recursion [Bertsekas (2000)],

$$J_k(x_k) = \max_{u_k \in \Omega} \{ \mathrm{E}_{wk} \{ g(x_k, u_k, w_k) + J_{k+1}(f(x_k, u_k, w_k)) \} \}. \quad (1)$$

In this equation, the state at a time step k is x_k, the vehicle assignments (or control) are u_k, that come from a set of possible assignments Ω, such as: turn left, turn right, ascend, descend, or go straight. The stochastic elements are contained in w_k. The term $g(x_k, u_k, w_k)$ is the single-step gain. Note that $x_{k+1} = f(x_k, u_k, w_k)$, which is a function that produces the next state from the current state, control, and stochastic elements.

For ease of presentation for the remainder of this paper, let Φ_k represent the triplet $[x_k, u_k, w_k]$, noting that the index of each member is the same as that of Φ. In (1), $g(\Phi_k)$ represents the gain that can be had at the present time, and $J_{k+1}(f(\Phi_k))$ represents the predicted gain from future times.

By the recursion (1), in order to make the computations necessary to calculate the optimal cost to go at time k, the value must be computed for not only the current state, but also for all potential future states. However, performing this computation is often computationally infeasible in practice. Following the work in [Flint and Fernandez (2005)], the DP recursion is computed exactly out to some fixed horizon of r decision time steps, and then an approximation of the cost to go is substituted for the actual cost to go at that point. The future cost to go term is approximated by a value $\tilde{J}(\Phi_{k+r})$, i.e.

$$J_{k+r+1}(f(\Phi_{k+r})) \approx \tilde{J}(\Phi_{k+r}). \quad (2)$$

Such an approach is often called *approximate dynamic programming*. The optimization is then performed on the approximate dynamic-programming equation

$$\begin{aligned} J_k(x_k) = \max_{u_k \in \Omega} \{ & \mathrm{E}_{wk} \{ g(\Phi_k) \\ & + \max_{u_{k+1} \in \Omega} \mathrm{E}_{wk} \{ g(\Phi_{k+1}) \\ & + \ldots + \max_{u_{k+r} \in \Omega} \mathrm{E}_{wk} \{ g(\Phi_{k+r}) \\ & + \tilde{J}(f(\Phi_{k+r})) \} \ldots \} \} \}. \end{aligned} \quad (3)$$

The approximate cost to go can be created by letting

$$\tilde{J}(f(\Phi_{k+r})) := \left(\sum_{\forall t \in \Theta} S_t(\Phi_k) \right) K_u, \quad (4)$$

where $S_t(\cdot)$ is a scoring function that approximates future gain based on several factors of what is known about a target t. The value K_u is a scaling factor to ensure that the value is appropriate given the single-step gains. This scaling factor is very important to the performance of the algorithm: it balances how much the vehicles weigh present considerations with future gain. The scoring function can be constructed in different ways, depending on the insight and useful heuristics available in a given modeling case. For example, in [Flint and Fernandez (2005)], the scoring function is defined as the combination of three modular

functions, each of which contains information important to the future gain of the vehicles. Thus, the scoring function is given by

$$S_t(\Phi_k) : = \left(\sum_{\forall t \in \Theta} L(\Phi_k), I(\Phi_k), M(\Phi_k) \right). \quad (5)$$

The value $L(\Phi_k)$ is a length factor that determines how far away a potential target is. The function $I(\Phi_k)$ is an interference prediction factor or cooperation factor that attempts to predict the effect of other vehicles' actions on the environment. The function $M(\Phi_k)$ is a mission position factor. This is high if accomplishing the task puts the vehicle in a good position to complete the other elements of the mission. (For a more-detailed formulation, see [Flint et al. (2003b)], and [Flint (2005)].)

The environment in which the vehicles are searching is shown in Figure 1. Each vehicle is equipped with a sensor that can detect targets on the ground beneath it as it travels. In the scenario shown, the vehicles are searching for ten targets. The targets are given to the vehicles as a suspected location, i.e. a coordinate pair, shown as an 'x,' as well as an uncertainty region (in this figure, a 10 km by 10 km rectangle) that within which, if the target really exists, the target is guaranteed to be. The sensor on board a vehicle is modeled as a line 2000 m across that sweeps the ground as the vehicle advances, essentially creating, for each simulation time step, a rectangular footprint of size 2000 m by however far the vehicle carrying the sensor travels in that simulation time step. A target that falls within this footprint has some probability ρ of being detected. If detected, it has some probability of being classified correctly. A threshold for detection in both probability of existence and probability of classification is assigned such that for any target with probabilities above that threshold, the target is considered found and is removed as a further search candidate. The number of these targets found during the course of a search mission is used as the metric of success.

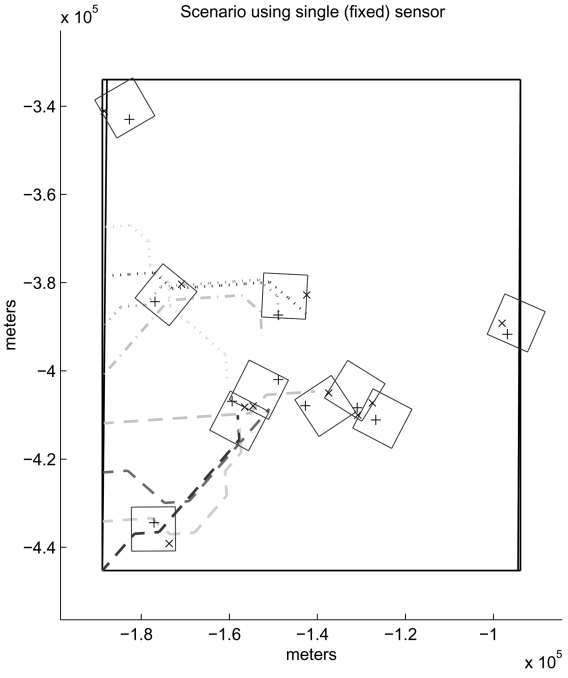

Figure 1. The simulation environment, showing several vehicles' paths.

The actual location of the target, which the vehicles do not know, is shown as a '+' in Figure 1. The targets and their uncertainty areas are placed randomly (with a uniform distribution over the the in-bounds region of the search environment) between each replication of the simulation. Aside from flat ground at an altitude of zero, which will cause the vehicles to crash if they contact it, there are no threats present.

THE RANDOM-NUMBER GENERATOR AND STREAM MANAGEMENT

Certain fundamental assumptions must be met for statistical methods to be employed in the analysis of simulation results. One of the most fundamental of these is the assumption that any trials or replications are independent of one another, or are at least dependent in a predictable manner (which the CRN method, given in the next section, will exploit). At the core of this assumption is the method used to generate the random numbers that are used to determine the stochastic elements that appear in the simulation. Unfortunately, not all random-number generators are statistically reliable, even those supplied with widely used software. For instance, the standard specification

for the generator in C++ (called rand ( )) allows a cycle length as small as 32,767, after which the random numbers will repeat themselves in exactly the same order [Kernighan and Ritchie (1988)]. Such a short cycle length is clearly insufficient for any serious simulation work. The danger of using a statistically questionable random number generator is that it can introduce a bias into the simulation results (by breaking fundamental statistical assumptions) that is extremely difficult to detect, and thus remove. This point is worth summarizing succinctly: the results of any simulation study using a bad random number generator may not be valid, and conclusions drawn from such studies may be incorrect. Furthermore, it will be nearly impossible to detect the problem until a new study is conducted using a different random-number source (e.g. real physics as in live flights against real targets); this may result in a very costly correction. In order for the results of a simulation experiment to be valid, every attempt to remove such bias must be taken, and using a good random-number generator (RNG) is essential to accomplishing this goal. For the simulation described in this paper, a combined multiple recursive generator (CMRG) (from the appendix of Chapter 7 of [Law and Kelton (2000)] and in [L'Ecuyer et al. (2002)]) is used. This random-number generator has proven good statistical properties. This CMRG is a good choice for this simulation, since the simulation is sufficiently complex to dominate any extra computation this generator requires over simpler, but less statistically well-behaved ones. It has a cycle length of approximately 3.1×10^{57}, more than enough to guarantee that each replication will be independent.

Another very useful characteristic of this RNG is the fact that it has 10,000 streams, or separate pools of random draws, that are spaced 10^{16} apart. Each stream is essentially a separate RNG, which uses a different seed from any other. This allows each source of randomness in the simulation to be assigned a single stream, which is then used for variance reduction, as discussed in the next section. As the first step in this process, every place in the simulation where random numbers are used is identified. For the particular example used in this paper, in each of these places some information is given showing for what they are used, the size of the vector in which they are used, the distribution used, the method for generating the distribution, and the stream index assigned to them. The streams are summarized in Table 2.

It was possible to use stream 1 for the first several draws because every replication used the same quantity of random numbers for these values, so that any commonness between the values was maintained. This is not true for the other values, since each was used a random number of times during the simulation. Somewhat generously, ten streams were allocated per replication so that future additions to the

Table 2. A description of the random numbers used in the simulation, along with information about the distribution, vector size, and stream assignment for each. Acc/Rej is short for Uniform with Accept/Reject, and is used to place targets in arbitrarily shaped 2-D regions.

Use Description	Distribution	Gen. Meth.	Size	Str #
Random placement of targets	2-D uniform	Acc/Rej	2	1
Random orientation of target	uniform over 0 to 360 degrees	Uniform	1	1
Placement of uncertainty area	2-D uniform within rectangle	Uniform	2	1
Guessed position (in uncertainty area)	2-D uniform within rectangle	Uniform	2	1
Resolve threat to vehicle	threshold comparison	Uniform	1	2
Sensor-detection check	threshold comparison	Uniform	1	3
Sensor-classification (true) check	threshold comparison	Uniform	1	4
Sensor-classification (false) check	threshold comparison	Uniform	1	5
Check for sensor false positive	threshold comparison	Uniform	1	6
Pick random track (for false positive)	discrete uniform	Uniform	1	7
Sensor error reject check	threshold comparison	Uniform	1	8

simulations might have a few spare streams. This meant that 1000 replications could be run before exhausting the 10,000 available streams. For each replication, there was assigned a base stream value (or stream seed) that was different for each replication. For each stream, the stream seed was added to the assigned stream index given in Table 2. On each subsequent replication, the stream seed was increased by ten, thus giving each replication an independent stream of random numbers. Thus, for example, in replication one, threat resolution uses stream 2; in replication two, the threat resolution uses stream 12; and so on.

VARIANCE REDUCTION

In the course of simulation experiments on the cooperative UAV problem, it is possible (and, in fact, common, as shown in the motivational example given in Table 1), that the randomness of certain elements in the scenario would dominate the results, such that the difference in the success metric between any two scenarios under test was small enough not to be statistically significant. Thus, the effect of what was being tested was lost in the noise of the random environment. For example, one factor that tended to dominate the performance of the search team being studied was the random placement of the targets at the start of the search. If the targets were randomly placed in areas that made them ''easy'' to find, then any algorithm would perform well, and if the targets are placed in areas that made them ''hard'' to find, no algorithm would perform well. The variance was such that the brute-force method of just running more and more replications of each scenario was becoming impractical, in that it was very difficult to get statistically significantly different results in a reasonable amount of time, since the simulation was complex enough that it took a non-negligible amount of time to run replications. This is compounded by the fact that a reduction in the statistical uncertainty requires a quadratic increase in the number of replications required, (e.g. to get double the statistical precision it would take quadruple the number of replications). In order to do the comparisons in a statistically valid manner while at the same time being able to collect results in a reasonable amount of time, the common-random-numbers (CRN) method of variance reduction was employed (see Chapter 11 of [Law and Kelton (2000)]).

In this method, any two scenarios in which a comparison was desired are run using the same streams of random numbers. This induces a positive correlation between the results, such that any differences in the performance during the scenario are much more likely to arise because of the item under test than because of a difference in the random numbers. This allows the confidence intervals around the mean performances to be shortened, such that statistically significant differences between scenarios can be shown. However, it is possible that the method will ''backfire,'' or produce larger confidence intervals, under certain conditions. Given that a complete simulation study generally takes a fairly large amount of time to complete, it is desirable to know in advance if any backfiring would occur. This is so that if it does occur, one would know not to waste time setting up a CRN scheme for the large study. Thus, a smaller pilot test was conducted first, using the CRN technique. If that test showed the desired improvement in simulation efficiency, the method would then be employed to the larger and longer simulation study. This is a good practice to follow in any complex simulation study.

PILOT TEST

The pilot run consists of ten replications with three distinct random-number stream management methods, meanwhile varying a parameter that is known to be directly proportional to a success metric (in this case, average performance of the search). A replication consists of running the simulator for 3000 simulation seconds, while recording the number of targets found per time step, where the time steps occur every 30 seconds. This provides 100 data points for each replication. The three random-number stream management methods are: (1) using the clock to seed the random-number generator, (2) using the same

stream for the entire simulation, and (3) using the streams assigned in the previous section to implement the CRN system. The parameter being varied was the sensor-detection probability (defined as ρ in [Flint et al. (2004a)]). The higher this number, the better the sensor, and therefore, the better the average performance of the search should be. This value is the threshold against which the random numbers generated from stream 3 are compared. Three classes of sensor are defined for this test. Each of the probabilities shown is conditioned on the sensor having the opportunity to detect the target.

Good: $\rho = 80\%$ chance of target detection
OK: $\rho = 70\%$ chance of target detection
Bad: $\rho = 50\%$ chance of target detection

Using the clock to seed the random-number generator is somewhat dangerous, statistically speaking, since it is possible that the same seed will be generated as in a previous trial. This means that the replication will be exactly the same as the previous trial, and the inclusion of such duplicate replications can bias the final results. However, for this experiment, steps were taken to ensure that this did not happen. Thus, each trial has a different seed and is completely independent of all others. The one-stream-per-replication method can induce some amount of correlation between results, in that if there is a fixed number of random draws at the start of the simulation, there will be some common random numbers between compared scenarios. The correlation will not be as strong as in the CRN case, though, where there are multiple, synchronized streams used.

A summary of the results of the pilot test is given in Table 3. The value given for each style of stream management and trial is the number of time steps at which the paired-*t* test found a statistically significant difference at a 95% level of confidence between the two sensor trials. The paired-*t* test had 9 degrees of freedom with the null hypothesis being that the means were equal. The method performed similarly when the difference between the scenarios (i.e. Good vs. Bad) was pronounced, but the CRN method did have a larger effect. But, as the difference in scenario lessened (i.e. Good vs. OK), the power of CRN is shown to be much more effective than the other methods of random-number selection. Thus, it does not appear to be backfiring for this type of simulation, and it would seem to provide some benefit in a more sophisticated and larger simulation study.

Table 3. Pilot Test (10 replications): Number of time steps where simulation (correctly) detected $\alpha = 0.05$ level of statistical difference.

Type	Good vs. Bad	Good vs. OK
Clock Seed	32	1
1 Stream / Rep	31	0
CRN	38	32

USING COMMON RANDOM NUMBERS IN A LARGER SIMULATION STUDY

Following the definitions in [Flint et al. (2004a)] and [Flint (2005)], unless otherwise noted, the sensor parameters are: the probability of accepting real detections is $\rho = 0.8$; the probability of rejecting false classifications is $\omega = 0.95$; the probability of accepting real classifications is $\psi = 0.8$; and the probability of rejecting false detection is γ, which is set such that one false positive among the entire team occurs on average every 60 simulation seconds.

A small designed experiment examined the sensitivity of the search mission to the future gain weight factor (K_u as defined in Equation (4)). The values of K_u tested were 0.02, 0.002, 0.0002, 0.00002, and 0. These values were chosen to give a broad spectrum of possibility for the optimal setting for this parameter. The factor of 0 forces the algorithm to ignore completely the future component, and focus solely on the shorter term. Each scenario is run through 500 replications. The metric continues to be the number of targets found by the team during the search mission. The results of the simulation study are summarized in Table 4 and Figure 4. Some results are also shown graphically in Figures 2 and 3. These figures show two of the comparisons made in Table 4. In Figure 2, the $K_u = 0.02$ case is shown to be inferior to the $K_u = 0.0002$ case by a statistically significant amount in the shaded region. In

Table 4. Summary of simulation study, using K_u as defined in Equation (4). Arrows indicate a result that shows $\alpha = 0.05$ level of statistical difference in favor of the value pointed to by the arrow. Approx. equal sign shows that there was not statistically significant difference. Numbers in parentheses indicate number of time steps where significance was detected (out of 100).

K_u	0.02	0.002	0.0002	0.00002
0.002	$\Leftarrow$ (91)			
0.0002	$\Leftarrow$ (82)	$\approx$ (0)		
0.00002	$\Leftarrow$ (87)	$\approx$ (0)	$\approx$ (0)	
0	$\Leftarrow$ (81)	$\Uparrow$ (48)	$\Uparrow$ (70)	$\Uparrow$ (72)

Figure 3, the cases are too similar to find any statistically significant difference at all, which is not surprising considering how similar the results appear.

From the results, it is clear that the choice of the future weight gain factor can significantly affect the quality of the search mission. Weighting the future too heavily can have a negative impact on the results, since the value for 0.02 is worse than the result for 0.002, 0.0002, and 0.00002. In fact, it would be better to ignore the future altogether in this case than to weigh it too heavily, as shown by the fact that the 0 case is better than the 0.02 case. This makes sense if one considers the fact that if the

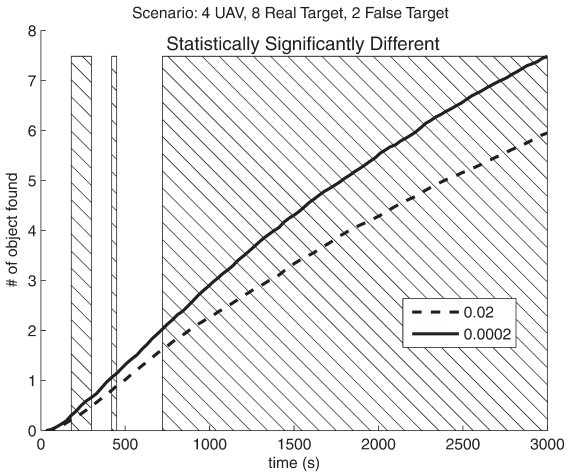

Figure 2. $K_u = 0.02$ and $K_u = 0.0002$ trial results; there is a significant difference in the shaded regions.

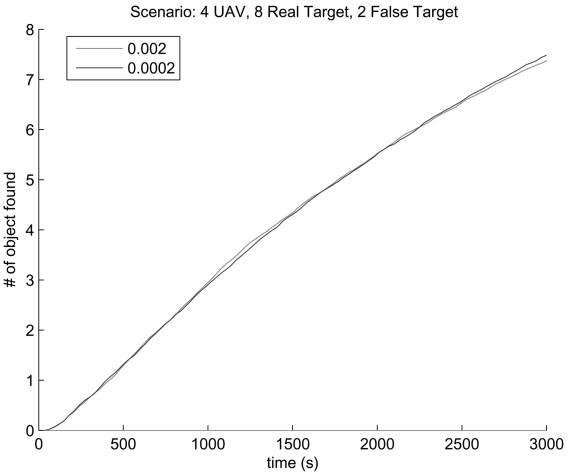

Figure 3. $K_u = 0.002$ and $K_u = 0.0002$ trial results; there is no significant difference anywhere.

future is considered too heavily, then it dominates actions in the present. In this case, because the future actions are more coarse than the present actions, there is always a future in the current model, and because realized gain can be collected only by completing present tasks, the overall performance suffers.

However, including a moderate value of K_u is shown to be superior to not including it. From the results, it would seem that the optimal value is in the range $0 < K_u \leq 0.002$. A more exact value could be found by conducting additional experiments.

In order to be confident in this conclusion, a good practice would be to check the data to ensure that CRN did not backfire. The data used for the 0 vs. 0.02 simulation were manipulated such that the paired-t test was conducted using independent trials, rather than with

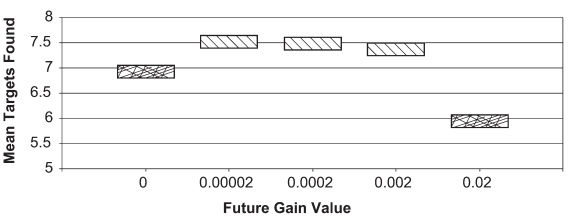

Figure 4. This shows how each of the scenarios compares to each other one, using the mean number of targets found out of 8. In this case, there are three that are statistically not distinguishable from one another (the top three) after 500 replications at $\alpha = 0.05$.

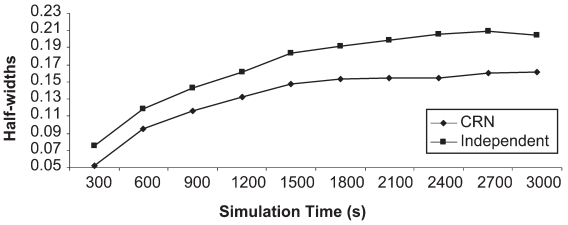

Figure 5. Half-widths at $\alpha = 0.05$ level of confidence of the paired-t test at various times during the $K_u = 0$ vs. $K_u = 0.02$ simulations.

CRN. This was done by pairing each replication result from one test with a replication result other than the correlated one, so that each replication result was compared to a replication with a different stream. Since a good RNG was used, there is no correlation between numbers from different streams and there is independence between the two runs. Figure 5 shows that CRN does indeed help, since this shows that the half-widths are always smaller in the CRN case than in the independent case. This is further reinforced by the results in Table 5, where using the same number of replications (and in fact, the same data,) it is possible to be more confident of the difference between the two averages in ten additional time steps in the CRN case. So, it must be that the CRN does not backfire, and in fact makes a helpful difference.

CONCLUSIONS

Given the complicated nature of multiple-UAV control algorithms and the fact that it is currently much easier to analyze and tune algorithms in simulation, simulation is a vital tool. Several simulation-analysis methods were studied in this paper. These include: a good random-number generator, streaming of random variables, and the common-random-numbers variance-reduction scheme. These were applied to the particular problem of discovering at what value the future gain weight factor should be set in an approximate dynamic-programming solution to a cooperative UAV search problem. This problem illustrates the potential for these methods to assist greatly in making statistically safe conclusions, not only for this problem, but for many others as well. Toward this objective, several good-practice guidelines were identified so that the work presented here can serve as a guide to others in their simulation studies. This allows new and effective courses of action to be identified and chosen such that the development of UAV control algorithms is made much more effectively, efficiently, and reliably.

Table 5. Checking for backfire: Number of time steps where simulation detected $\alpha = 0.05$ level of statistical difference.

K_u	0 vs. 0.02
CRN	81
Independent	71

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers of this paper for their helpful comments and suggestions. We feel that the paper is much stronger as a result of their time and efforts.

REFERENCES

Bertsekas, D. P. 2000. *Dynamic Programming and Optimal Control,* vol. 1. 2nd ed. Athena Scientific, Belmont, Massachusetts.

Cassandras, C. G., W. Li. 2002. A receding horizon approach for solving some cooperative control problems. *Proc. of the 41st IEEE Conference on Decision and Control.* Las Vegas, Nevada, USA, 3760–65.

Castanon, D. A., C. Wu. 2003. Distributed algorithms for dynamic reassigment. *Proceedings of the 42st IEEE Conference on Decision and Control.* Hyatt Regency Maui, Hawaii, USA, 13–18.

Flint, M. March 2005. Cooperative unmanned aerial vehicle (UAV) search in dynamic environments using stochastic methods. Ph.D. dissertation, electrical engineering, University of Cincinnati.

Flint, M., E. Fernandez. 2005. Approximate dynamic programming methods for cooperative UAV search. *Proceedings of the 16th IFAC*

World Congress, vol. Volume A4. Prague, Czech Republic, 59–64.

Flint, M., E. Fernández-Gaucherand, M. Polycarpou. 2003a. Cooperative control for UAV's searching risky environments for targets. *Proc. of the 42nd IEEE Conference on Decision and Control.* Hyatt Regency Maui, Hawaii, USA, 3567–3572.

Flint, M., E. Fernández-Gaucherand, M. Polycarpou. 2003b. Stochastic modeling of a cooperative autonomous UAV search problem. *Military Operations Research* **8** 13–32.

Flint, M., E. Fernández-Gaucherand, M. Polycarpou. 2004a. Efficient Bayesian methods for updating and storing uncertain search information for UAV's. *Proc. of the 43nd IEEE Conference on Decision and Control.* Atlantis, Paradise Island, Bahamas, 1094–1098.

Flint, M., E. Fernández-Gaucherand, M. Polycarpou. 2004b. A probabilistic framework for passive cooperation among UAV's performing a search. *Proc. of the Sixteenth International Symposium on Mathematical Theory of Networks and Systems (MTNS2004).* Leuven, Belgium.

Flint, M., M. Polycarpou, E. Fernández-Gaucherand. 2002. Cooperative control for multiple autonomous UAV's searching for targets. *Proc. of the 41st IEEE Conference on Decision and Control.* Las Vegas, Nevada, USA, 2823–28.

Hespanha, J. P., H. H. Kizilocak. 2002. Efficient computation of dynamic probabilistic maps. *Proc. of the 10th Mediterranean Conf. on Control and Automation.* Lisbon, Portugal.

Jin, Y., A.A. Minai, M.M. Polycarpou. 2003. Cooperative real-time search and task allocation in UAV teams. *Proceedings of the 42st IEEE Conference on Decision and Control.* Hyatt Regency Maui, Hawaii, USA, 7–12.

Kernighan, Brian W., Dennis M. Ritchie. 1988. The C Programming Language. 2nd ed. Prentice Hall PTR, Upper Saddle River, NJ.

Law, Averill M., W. David Kelton. 2000. *Simulation Modeling & Analysis.* 3rd ed. McGraw-Hill, Inc, New York.

L'Ecuyer, P., R. Simard, E.J. Chen, W.D. Kelton. 2002. An object-oriented random-number package with many long streams and substreams. *Operations Research* **50** 1073–1075.

O'Rourke, K. P., W. B. Carlton, T. G. Bailey, R. R. Hill. 2001. Dynamic routing of unmanned aerial vehicles using reactive tabu search. *Military Operations Research* **6** 5–30.

Richards, A., J. Bellingham, M. Tillerson, J. How. 2002. Co-ordination and control of multiple UAVs. *Proc. of the 2002 AIAA GNC Conference.* Monterey, California.