



Stochastic Agent-Based Analysis of UAV Mission Effectiveness

Philippe Ranque¹, Dane Freeman², Kemp Kernstine³, Dongwook Lim⁴, Elena Garcia⁵, and Dimitri Mavris⁶
Aerospace Systems Design Laboratory
Georgia Institute of Technology, Atlanta, GA 30332-0150

The investigation of stochastic agent based simulations adds new difficulties to exploring design spaces and analyzing the results. These simulations no longer fulfill the requirements of deterministic computer simulations, and yet require fewer assumptions than traditional design of experiments; however in either exploration enormous amounts of data must be understood. This paper starts by developing a family-of-systems UAV simulation to investigate varying techniques of understanding the data from these simulations.

I. Introduction

A. Motivation

1. Aircraft Family

When considering a range of different products to satisfy a set of target market niches it is important to consider the set of products as a whole.¹ The product family can then be designed to minimize the duplication of effort by maximizing commonality at the product component level. Determining the opportunities for component sharing is critical to designing a well performing product family. If the components are too dissimilar then sharing components can compromise both products' performance. Identifying those components that are similar enough is a difficult combinatorial problem and made more complex when coupled with larger system-of-systems problems.

This paper looks at enabling a larger product family design methodology by first starting with the development of the system-of-systems environment and identifying possible ways to create surrogate models necessary for a more in depth exploration of the family design space. To conduct this research a SoS simulation using a common top-down approach in aircraft design was built. This approach starts with top level operational effectiveness requirements followed by a SoS simulation to set mission performance requirements at the vehicle level. This SoS modeling is used to create a mapping from vehicle capabilities, known as Measures of Performance (MoPs), to overall mission-level Measures of Effectiveness (MoEs).

2. System-of-Systems

In the 1970s the first instance of system-of-systems arose in Ackoff's work² when he described the current influence of systems on their surroundings systems. This basic principle of interacting systems is the fundamental concept behind SoS. It has been known for some time that SoS are not well handled by traditional system engineering methods,³ but it was not until 1981 that Blanchard and Fabycky introduced "system-life-cycle engineering" to engineer open systems.⁴

In the last 30 years the field has grown significantly,⁵ with its first engineering application in the Strategic Defense Initiative in 1989.^{4,6} However, even with this growth, the field is still in its infancy with many definitions⁷⁻¹⁰ circulating in the literature. Despite the literary variation in definitions the field seems to have converged on the overall features which describe a SoS. Although there is still some variation in the precise words chosen, the underlying concepts are uniform. It is typically Maier's¹¹ characteristics which are seen as the basis for analysis. (Boardman et al.¹² conducted a literature review of SoS definitions which further established Maier's characteristics

¹ Graduated, School of Aerospace Engineering, 270 Ferst Drive, AIAA member.

² Ph.D. Student, School of Aerospace Engineering, 270 Ferst Drive, AIAA student member.

³ Ph.D. Student, School of Aerospace Engineering, 270 Ferst Drive, AIAA student member.

⁴ Research Engineer II, School of Aerospace Engineering, 270 Ferst Drive, AIAA member.

⁵ Research Engineer II, School of Aerospace Engineering, 270 Ferst Drive, AIAA member.

⁶ Boeing Professor, School of Aerospace Engineering, 270 Ferst Drive, AIAA Fellow.

as the most influential.) From Boardman et al. these characteristics are: autonomy, belonging, connectivity, diversity, and emergence. For extended definitions the reader is directed to references.¹¹⁻¹²

3. Analysis Challenges and Objectives

Most of the characteristics are easily captured in various simulation environments; specifically autonomy belonging connectivity, and diversity. However, it is the last, emergent behaviors, which can be difficult to capture in many simulating environments. For this reason an agent based simulation was chosen for this study.

Additional challenges arise in investigating the data of SoS simulations. SoS simulations are a collection of controlled variables (like wing size or fuel quantity) and random noise variables (like starting location or weather pattern). Meilich¹³ identified several additional issues when addressing SoS architectures like when is the simulation good enough, and how much one can trust the simulation. The answer to these questions is that it is not the exact output of the simulation that is desired, but an approximation to the trend: an investigation of the impacts. From this it can be surmised that although exact solutions are ideal, a great amount of information can be gained from approximations.

An example of these approximations is finding where individual system level optimums begin to break down. It is known that in systems, and thus in their bigger brother SoS, the optimums amongst the constituent systems may not produce an overall optimum.¹⁴ The individual optimums could actually cause systems to compete.¹⁵

Finally, in analyzing these SoS, additional issues are encountered from their implicit statistical attributes. To calculate the variability at each design location, a Monte Carlo simulation is conducted with differing random variable settings. This distribution can change for every design point in the simulation creating a heteroscedastic space, and limiting inferences from regression methods. If this complication by itself is not enough, these spaces are further correlated and are of very high dimension.

Agent based simulations typically require a long run time, especially as they need to be repeated n times per design point. This motivates the need for creating surrogate models in order to infer responses in areas that were not investigated. Techniques used for surrogate modeling, however, are usually based on deterministic simulation results and are not suitable for a stochastic model.

B. Example Problems

The agent based SoS modeling approach was demonstrated on two problems (cf. Figure 1). The first problem deals with aerial firefighting in the remote Greek islands and the second consists of monitoring maritime traffic and oil platforms off the coast of Norway. Each of these examples includes a set of very different missions ranging from long endurance missions requiring good sensors to time-critical delivery missions. A variety of missions in each scenario was desired to exercise the concept of a UAV family design.

The remote southeastern Greek islands experience frequent rapid summer forest fires as the land is dry and exposed to wind. Furthermore, these islands are mountainous and far from the mainland which makes ground-based firefighting difficult. Hence, the concept of using a set of UAVs for long endurance patrols above these islands during risky days in conjunction with a set of UAVs capable of dropping fire retardants.

Maritime traffic is very dense in the North Sea, especially in the south of Norway near the strait leading to the Baltic Sea. Also, several Norwegian offshore oil platforms lie in the same area, around 70 nm off the coast. These platforms require persistent surveillance by the coast guard to monitor the release of oil into the sea as well as protect the rigs themselves. The problem consists in designing a family of UAVs suitable for maritime surveillance, Search and Rescue (SAR) of distressed ships, and environmental monitoring. Having a persistent aerial surveillance allows the tracking of polluters such as tankers cleaning out their tanks, monitoring of trawlers to ensure they stay in the fishing areas, and eventually to prevent terrorist attacks on the platforms. All these missions were included in the

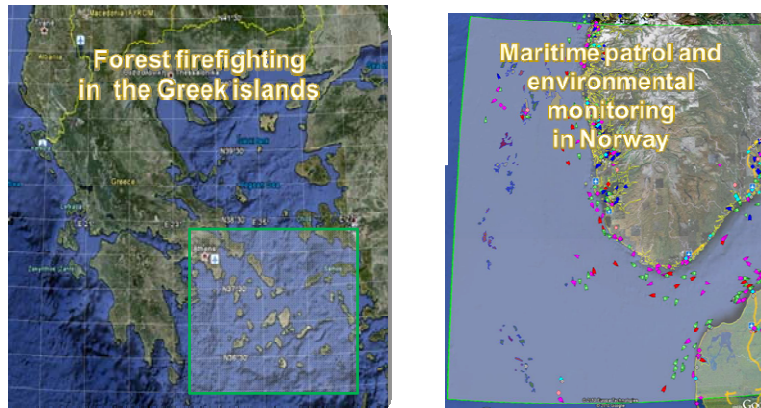


Figure 1. Location of the two example problems (©Google Earth).

simulation in addition to more time-sensitive missions involving payload dropping such as SAR or oil spill abatement.

II. Approach

A common approach to analyze stochastic simulations is to use a traditional Design of Experiment (DoE)¹⁶ such as a Latin hypercube¹⁷ to populate the design space and then run the simulation for several repetitions at each design point to capture the variations due to the random variables.¹⁸ For SoS simulations, one design point represents a certain combination of vehicles with given characteristics.





Design Point	Inputs (MoPs)			Outputs (MoEs)	
	Input1	Input2	Input3	Ouput1	Ouput2
1	Design of Experiment				
2					
...					

Figure 2. Input Design of Experiment (DoE) and output distributions.

At each design point, repetitions might be run until it is judged that there is enough knowledge about the output distributions. Hence the number of runs depends mostly on the *parameters of interest* in the distribution, the *confidence interval* needed and the *shape* of the distribution. If the parameters of interest are well defined (means, quantiles, etc.) it is possible to stop repeating the simulation once a satisfactory confidence interval is obtained. The objective of the study was not to optimize that number of repetitions, but rather to see how results could be analyzed assuming that there were enough repetitions.

With the SoS field in its embryonic stage, there is very little literature on the best practices for analyzing these stochastic computer simulations. Often when discussing the exploration of design spaces in computer simulations readers are turned to Sacks et al.,¹⁹ but the underlying assumption is that the simulations are deterministic. Although these simulations are deterministic for a given set of random variable settings, it is desired to capture the influences of these random variables. For stochastic simulations two common practices exist, using crossed arrays²⁰ or combined arrays.²¹ A common practice in the literature is to use a combination method where all random effects are lumped together and many repetitions are run of identical DoEs to capture the random influences (effectively a Monte Carlo at each design point). There is still significant research to be explored in the proper analysis of these environments. The following analysis aims at comparing different approaches to fit a surrogate model on such stochastic results.

1. Problem complexity

For a deterministic output, customer requirements are usually translated to *make sure that the output is either smaller/larger than a certain value or that it is optimal with respect to a criterion*. In the case of a stochastic output, it is harder to define the customer need. For instance, in the firefighting simulation, good design points would be those that *make sure that most of the fires remain small and make sure that they never get too big*. It means that when looking at the distribution of the size of burnt land, it is preferable to have *most* of the fires *small* and ensure that the *tail* of the distribution is *not too long*. It would not be enough to track the mean burnt land, nor would it be sufficient to control the 90th percentile (the size which the fire is smaller than 90% of the time). This supports the idea of capturing as much of the distribution as possible in the surrogate model.

In addition, some distributions are more complex than others. For instance, if the output of interest has a normal distribution the mean and the standard deviation would be enough to describe it. The following study focuses on the more general case, when there is no obvious simplification of the distribution. The simulation outputs support this decision since the distributions obtained are typically complex.

2. Distribution parameterization techniques

Creating a surrogate that would capture the shape of the distribution means finding a way to decompose the distribution. This decomposition should be the same on the entire design space. Then standard surrogate modeling techniques could be applied to interpolate the parameters between design points. The following techniques can be considered: law fitting, quantile decomposition, moment decomposition, wavelet decomposition,²² and quantile regression.

Law fitting assumes that distributions keep a similar parametric shape over the entire design space. The chosen type of parametric distribution has to fit the data properly at each design point. Finding the proper shape to be

applied across the design space may be difficult, but if the parametric distribution fits well, this method could be very powerful and use a small number of sample points, i.e. fewer repetitions of the simulation.

A distribution can also be described by a set of quantiles as shown in Figure 3. This has the advantage of not assuming a particular shape for the distribution. However, to get a good precision on the entire Cumulative Distribution Function (CDF) the set of quantiles should be carefully chosen and take into account the shape of the distribution.

Moment and wavelet decomposition consist in expanding the distribution as a generalized Fourier series. The problem consists in finding a suitable basis for this expansion. As there are an infinite number of terms in a generalized Fourier expansion, an appropriate choice of basis approximates well a large set of possible responses with only few terms of the expansion employed. For instance, moment series or wavelet series can be used. Describing a distribution by some of its moments (mean, standard deviation, skewness, kurtosis, etc.) might make sense if the distribution fits a

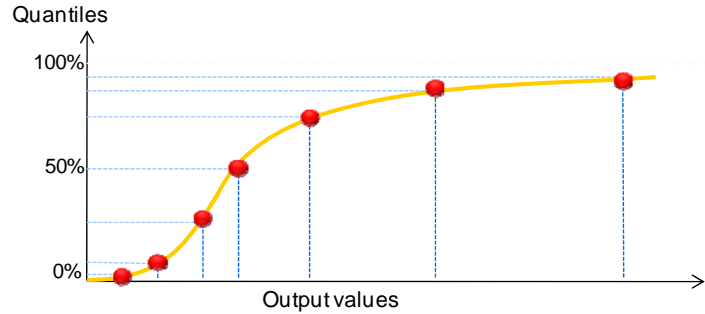


Figure 3. Quantile decomposition of a CDF.

standard shape. In particular, it works well for normal distributions. The wavelet series would be another solution, as it allows a parsimonious expansion for a wide variety of functions, including inhomogeneous cases. Wavelets were not further explored in this study but would be a very interesting field to investigate.

Quantile regression is a modeling technique where the conditional quantiles are estimated from the data. This is done by formulating an optimization problem and solving for a minimized loss function constraints can be implemented to ensure the resulting quantiles do not cross. Using a set of quantile models it is possible to gather more detailed information about the response distribution. Unlike other modeling techniques which regress to the mean, this is a more robust modeling technique because outliers have less effect on the median than the mean.²³ There are a few recent developments that use nonparametric quantile regression to model heteroscedastic spaces.²⁴ This allows for true exploratory data analysis where no statistical model has yet been defined. Currently work is being done in this area to allow its application to large data sets.

III. Modeling

A. Problem Definition

1. Mission decomposition and grouping

The two simulations were designed to formulate system requirements for a UAV family. Thus, the first step in defining both problems was to identify the functional requirements of each scenario.

In order to capture the concept of an aircraft family, at least two different aircraft types are needed for each problem, where each type has different requirements. If the two aircraft types perform too similar of missions, the analysis would lead to very similar designs and the family design problem would be less significant. Also, as a benchmark for the family design, it had to be possible to see how well a single aircraft type could perform all the missions. This is important to get a real measure of the benefit of designing a family.

For the firefighting problem, the missions are decomposed as follows:

- 1) Fire patrol: Patrol the Greek islands and quickly detect fires.
- 2) Fire monitoring: Provide awareness of its location, size and evolution.
- 3) Fire response: Drop retardant at the edge of the fire. Possibly drop retardant ahead of the fire.

The first two missions are long endurance persistency missions and the third is a time-sensitive dropping mission which naturally leads to using two different types of aircraft.

For the maritime problem, the missions are decomposed as follows :

- 1) Persistent patrol: Patrol the Norwegian Exclusive Economic Zone and provide awareness to the Norwegian Coast Guard. It includes:
 - Ship tracking: Aid the Automated Identification System (AIS) and the new Long Range Identification and Tracking (LRIT) system by monitoring and verifying ship location
 - Fishery patrol: Identify fishing vessels in protected zones and determine activity
- 2) Environmental monitoring: Quickly detect and verify oil slicks and collect sufficient evidence for legal action²⁵ (i.e. take a picture of the polluter with its spill).

- 3) Search: Locate distressed vessel and people. Monitor the situation until additional assistance arrives.
- 4) Spill response: If an oil slick is appropriately large, deliver tools quickly to combat it.
- 5) Rescue: Deliver emergency aid to distressed vessel and people. Monitor the situation until additional assistance arrives.

The first three missions can be classified as long-endurance reconnaissance missions and the last two as time-sensitive delivery missions. These two groups have different mission profiles, as shown in Figure 4, and different types of payload. Reconnaissance aircraft might have a large aspect ratio and mostly carry fuel and good sensors whereas delivery aircraft might be larger and able to carry large payloads on short distances. Consequently, two types of aircraft were used in the simulation: one for reconnaissance and the other one for delivery.

Both problems include two types of aircraft with conflicting requirements. The performance of both aircraft types is given as an input. It is also possible to use only one aircraft type to perform all the missions. It should be noted that performance inputs to the simulation are assumed to be internally consistent as no aircraft sizing routine is included in the SoS simulation.

2. Initial conditions

In order to save CPU time, disturbing events such as spills, distressed ships, or fires are triggered at time zero. As both simulations are easily reconfigurable, it is easy to vary the number and location of these disturbing events. However, to simplify the exploration of the models, their number is fixed and their location is random. Two spills and two distressed boats are triggered in the maritime simulation. One fire is triggered in the firefighting simulation.

In reality, events can occur at any time of the day, and at any moment of the patrol. As they are created at time zero in the simulation it is important to initialize the simulation as close to steady state as possible. This means having patrol aircraft already in the air with a random amount of fuel and assigning other aircraft randomly to airfields. Also, for the maritime simulation, the boat initialization is more complex as the *time between two ship detections* is a tracked MoE. With a bad initialization, the *time between two detections* distribution takes time to converge. Therefore, a simple surrogate based on the number of patrol aircraft, their cruise speed, and their detection radius is used to initialize the simulation properly.

In the firefighting problem, some aircraft patrol until the fire is found, then an aircraft is sent to monitor the fire and airtankers are sent to drop retardant to extinguish the fire.

In the maritime problem, some aircraft patrol to find oil spills, and then attempt to take a picture of the polluter while a delivery aircraft is sent to drop dispersant on the spill. When there is a ship in distress, the search mission is performed until the ship is found, then a rescue aircraft is sent to drop a lifeboat.

Both simulations stop when all the missions are accomplished or after 24 hours. This assumption was used because a system of aircraft that takes more than 24 hours to accomplish the missions is probably not a system of interest.

3. Measures of Performance (MoPs)

In this paper, the goal of modeling SoS missions is to set requirements on vehicle performance by evaluating system effectiveness within operational environments. These simulations create a mapping from the system measures of performance (MoPs) to the SoS measures of effectiveness (MoEs). Each simulation involves two types of aircraft, and for each aircraft the following MoPs are given as inputs: aircraft fuel efficiency, lift to drag ratio, cruise speed, operating empty weight, fuel capacity, and payload weight. The mission analysis was performed using Breguet Range Equation. Two types of the sensors—radar and IR camera—were considered. For the radar, detection angle and range to detect smoke (fire mission) or ships (maritime patrol) were treated as independent variables. One type of IR camera with two different operating modes, either short range with high resolution or long range with low resolution, was assumed. The delivery aircraft was assumed to carry different types of payload for each mission depending on the needs. The types of payload considered in the two simulations are fire retardant, oil spill dispersant, and lifeboat. The numbers of each type of aircraft were treated as independent variables. For the firefighting problem, the number of searching, monitoring, and extinguishing aircraft were user input. For the maritime problem, the numbers of delivery and surveillance aircraft are independent variables.

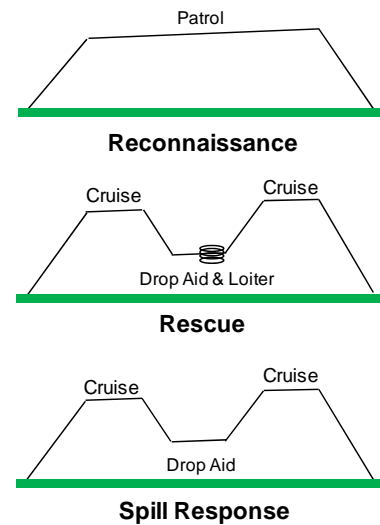


Figure 4. Maritime mission profiles.

4. Measures of Effectiveness (MoEs)

MoEs (Measures of Effectiveness) have been defined for each problem based on customer requirements. The simulation environments were designed in order to quantify them under each scenario. Hence they had a very important role in the design of the simulation: all the assumptions had to have an impact on the MoEs. Figure 5 gives the list of simulation outputs.

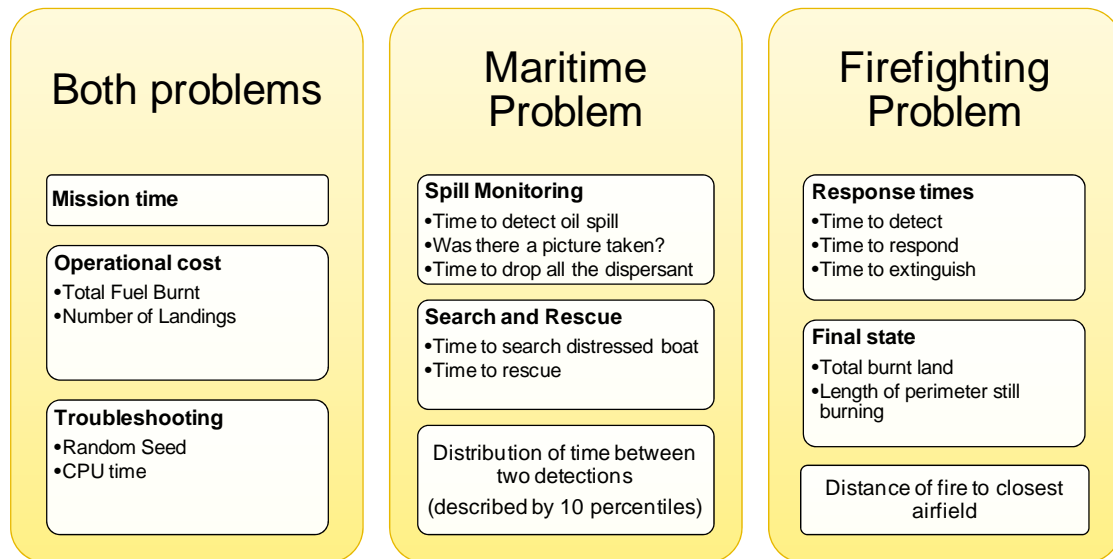


Figure 5. Outputs of both simulations.

B. Simulation Algorithm

Both simulations follow the algorithm described in Figure 6. As previously stated the initialization aims at simulating a steady state in order to minimize the simulation time. There is a time increment of one minute at each main loop. When the end condition is reached - mission finished or the 24-hour threshold - the MoEs are saved. The main steps are described below.

1. Mission allocation

For each mission, the number of aircraft needed is assessed. If there are more aircraft doing the mission than what is needed, some of them are asked to come back to an airfield. If there are not enough aircraft, the algorithm finds a group of aircraft compatible with the mission. A compatible aircraft meets the four following conditions: 1) they have the capability of performing this mission (sensors and payload); 2) they are not already doing another mission; 3) they currently carry the right payload type; and 4) they have enough fuel to do the entire mission.

The closest-to-target vehicle(s) is (are) then selected from this group of compatible aircraft. A priority order was defined to avoid conflicts between missions. For instance, search and rescue missions have the highest priority. If there is no available aircraft on the ground, it is possible to use aircraft doing patrol for another task, as the patrol does not have a high priority. In case there is no available aircraft carrying the right payload (e.g. lifeboat instead of dispersant) then one would be asked to change its payload on the ground.

2. Aircraft update

Depending on the new allocations and the new positions, each aircraft is updated in a random order. First, the mission status is updated: aircraft switch to another part of the mission or another mission if the current destination is reached or if a certain time is elapsed. Second, the mission segment is updated based on the next destination. Then the speed, the position, and the weights are updated. Finally, sensors detect the area seen from the new position and eventually see ships, spills, or fires that can trigger new missions.

3. Environment update

The firefighting simulation includes a model of fires. This model is a tradeoff between model accuracy and pixel size. With more pixels the simulation gets slower but it is easier to implement a fire model. A way of getting a good fire model without using too many pixels was to attribute each pixel a certain percentage of burnt land. The impact of wind and terrain conditions on the fire spread was taken into account using observations of reference.²⁶ The sum of these percentages gives the exact area of the fire which is used to get an estimate of the perimeter length, as aircraft extinguish a certain perimeter length when they drop retardant.

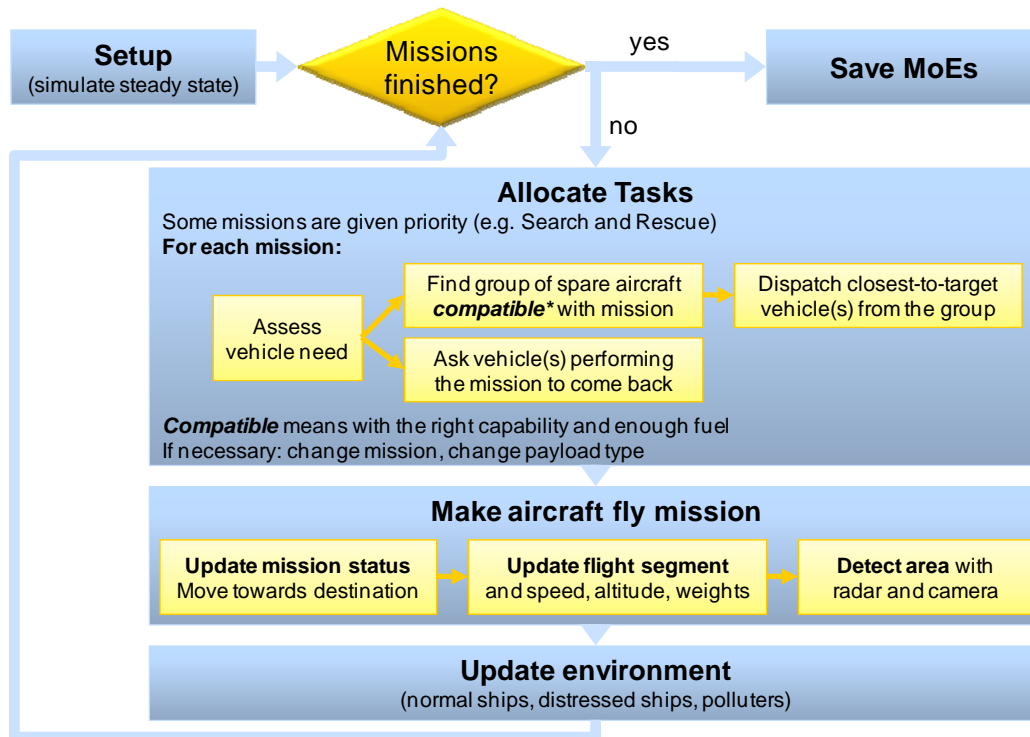


Figure 6. Simulation algorithm.

In the maritime simulation ships are given a very simple behavior: they are moving preferably towards shipping channels and turning if they have an obstacle in front of them. This very simple logic captures typical behavior and does not significantly slow down the simulation. Furthermore, the model was designed to be close to reality by using a representative number of each different type of ship, and by attributing them a speed accordingly.²⁷ Distressed boats have a different behavior as they are following a strong current (4kts, as it usually does not exceed 2 m/s in this region²⁸) with a random direction.

C. Simulation Environment

The agent-based simulations* were developed with Netlogo, an easy-to-use programmable modeling environment. Netlogo has its own programming language which is very intuitive and includes functions commonly used for agent-based approaches. This coding environment makes it possible to change the code ‘on-the-fly’ and to directly see the impact on the visualization environment, which is very convenient for troubleshooting.

Figure 7 and Figure 8 show both simulations’ display. Aircraft position, routes, and detection radiuses are intuitively displayed on a background map. Furthermore, the user can easily display the mission profile and the fuel weight graphs as well as access information about any agent by right-clicking on it, even while running the simulation. Measures of effectiveness and some monitoring variables are also displayed in real time. The UAVs’ and ships’ colors match their type and different symbols were used for detected hazards and their current state. Having such a good visual interface helps to make sure that the mission implementation is correct, e.g. check that the radar ground detection disk changes with the type of aircraft, the task and the altitude.

On the left side, buttons allow an easy setup of the environment, and scrollbars can adjust a subset of the input parameters. A DoE can be directly run from this interface by importing lists of input values, and outputs are then exported to a file.

D. Simulation Noise

As with most complex agent-based simulation, the two simulations created are stochastic. It means that when looking at a particular output, each combination of inputs will not have only one value but a distribution of values. The spread of these distributions depends on the randomness introduced in the simulation. For analysis purposes, it is important to understand what these internal random variables are. There are three different sources of noise for

* The simulations used for this paper can be found <https://ftp-asdl.ae.gatech.edu/nlogo> usr:nlogo pass:tui&

both SoS simulations: the randomness of the initialization, the update of agents in a random order, and the randomness of the environment model.

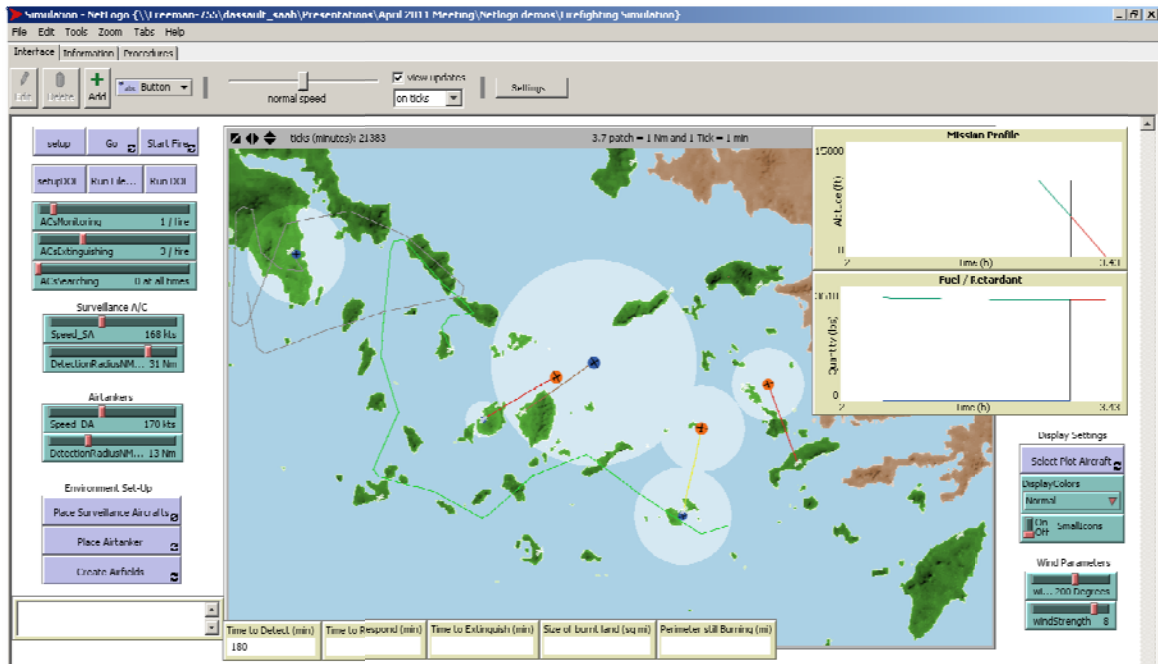


Figure 7. Firefighting simulation display.

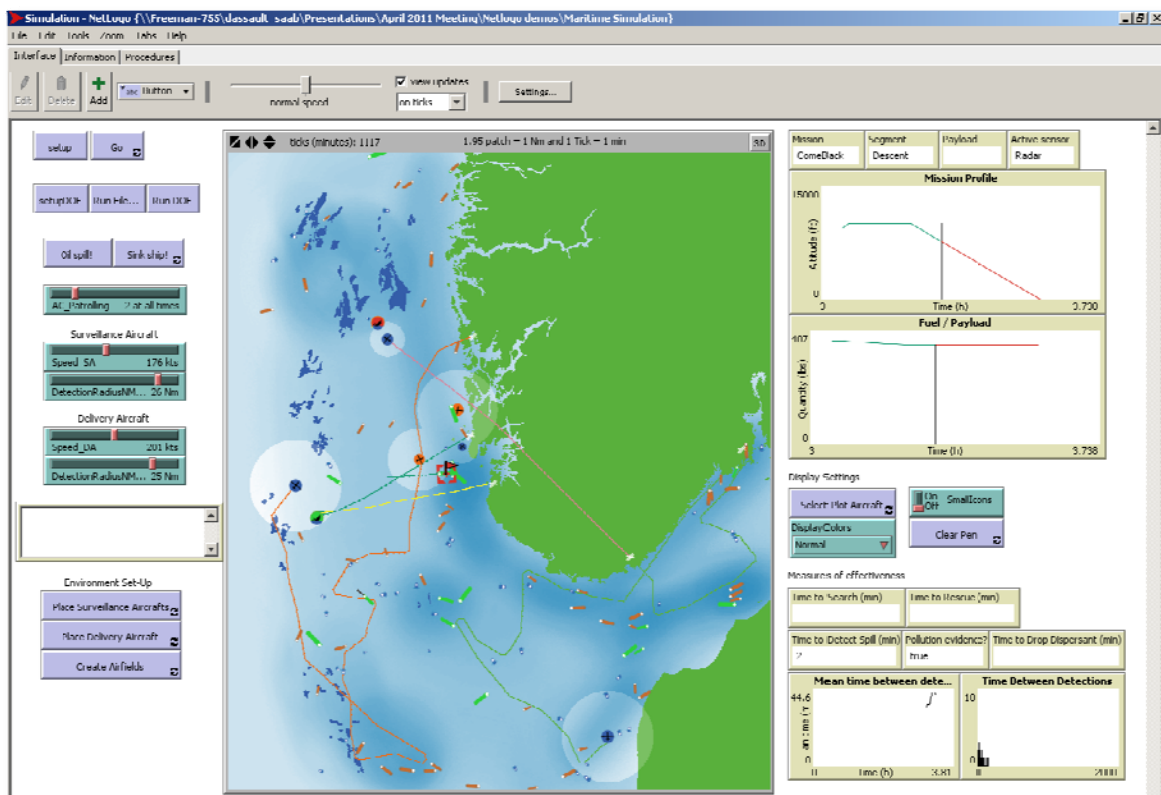


Figure 8. Maritime simulation display.

1. Initialization noise

The initialization is the principal source of randomness. For both simulations aircraft are randomly assigned to airfields. The location of the fire in the firefighting problem is random among the Greek islands. The initial position of the ships - especially the distressed ships - and the tankers polluting are random in the maritime simulation.

A way to understand the output would be to vary these initial positions intelligently instead of randomly, which would mean using these noise variables as input variables. For instance, it would be interesting to analyze the relationship between some of the noise variables such as 'distance of the fire to the closest airfield' or 'distance of a spill to the closest airfield' and some of the MoEs. If there is a significant relationship, it would help in fitting surrogates of the MoEs. However, as a first step, the initial positions were kept random.

2. Random order of agent update

In agent-based simulations and especially in the simulations created in Netlogo, each time a group of agents is updated they are called in a random order.²⁹ Indeed, if there is some interaction between two agents of a group, having always the same agent taking its decision first would lead to uncontrolled modeling biases. Calling them in a random order reduces the bias of using sequential discrete decision logic.

3. Environment model

There are random variables used in both problems to make the environment closer to reality. These have a minor impact on the outcome. The fire model is random as its speed oscillates around a fixed mean to make it more realistic. The ship traffic is modeled using agents following very simple rules. The ship behavior is random, however, the ship traffic tends to be in the same areas so the noise created by this is minor.

IV. Data Analysis

A 1200 point Latin hypercube design of experiments was used to determine which cases needed to be evaluated to best fill the interior of the design space. To try and capture the stochastic nature in the responses, 20 repetitions were performed for each design point for a total of 24,000 cases run. With each run completing in about a minute the total computation time was 400 CPU hours.

Table 1 shows the input ranges for the 12 input variables as well as the output variables. This analysis focused on the firefighting simulation and in particular on the analysis of one representative output: the size of the burnt land at the end of the simulation which was measured in hectares (100m*100m).

Table 1. List of inputs and outputs.

	Variable Name	Description	Low Bound	Upper Bound	Units
INPUTS	ACsExtinguishing	Number of delivery aircraft	2	8	
	ACsSearching	Number of search aircraft	1	6	
	Velocity_DA_Cruise	Speed of delivery aircraft	131.744	197.616	kts
	Velocity_SA_Cruise	Speed of search aircraft	131.744	197.616	kts
	L/D_DA_Cruise	Lift to drag ratio of delivery aircraft	22.12	33.18	
	L/D_SA_Cruise	Lift to drag ratio of search aircraft	22.12	33.18	
	FuelCapacity_DA	Fuel mass of delivery aircraft	400	4000	lb
	FuelCapacity_SA	Fuel mass of search aircraft	400	4000	lb
	Retardant Weight	Retardant weight	2000	12000	lb
	TSFC_DA_Cruise	Thrust specific fuel consumption of delivery aircraft	0.27712	0.48496	lb/hr/lb
	TSFC_SA_Cruise	Thrust specific fuel consumption of delivery aircraft	0.27712	0.48496	lb/hr/lb
	RadarRange_SA	radar range of search aircraft	37000	200000	meters
OUPUTS	Time to Detect	Time to detect a burning patch			min
	Time to Respond	Time to respond once the fire is detected			min
	Time to Extinguish	Time to put out fire			min
	Size of Burnt Land	Size of burnt land at end			hectares
	Perimeter Still Burning	Perimeter still burning at end			km
	Distance to Airfield	distance from closest airfield to fire			nmi
	Total Fuel Burn	Amount of total fuel burnt			lb
	Nb_Landings	number of landings			

1. Understanding the output

The size of burnt land was not completely captured as the simulation stops after 24 hours. Figure 9 shows the histograms and statistics of the simulation time and the size of the burnt land. Blue dots in the histograms represent

simulations where the fire was extinguished within 24 hrs. Red points are the simulations that reached the 24-hour limit. The histogram for the simulation time shown in the upper side of Figure 9 has a bimodal distribution. The distribution at the left side of the histogram represents the 90% of the 24,000 simulations that extinguished the fire within 3 hrs. At the other spectrum of the histogram is the other 10% of the simulations that failed to extinguish the fire within 24 hrs.

The histogram of the size of burnt land shown in the bottom of Figure 9 has a sharp peak at the low end and then a wide uniform distribution. For 90% of the simulations the size of the burnt land was kept below 25.1 ha. On the other hand, the fire grew as big as 8373 ha when the simulation failed to control the fire within the first couple of hours. This result shows that it was critical to put out the fire in 3 hrs or the fire got too big for the family of aircraft to control.

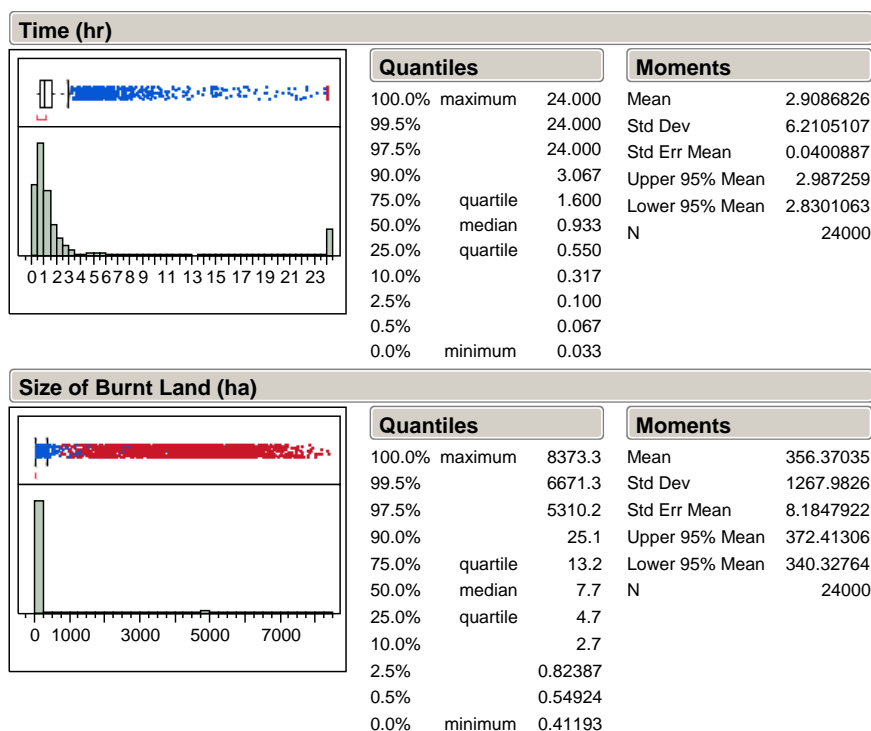


Figure 9. Histograms and Statistics of Simulation Time (Top) and the Size of Burnt Land (Bottom).

2. Recursive Partitioning Analysis

One very effective multivariate statistical analysis is recursive partitioning.³⁰ Partition analysis allows for identification of the key independent variables that have a significant effect on the output variable. For the key independent variables, the partition analysis creates groups of the data that best fit the output value. The outcome can be visualized by a tree-like structure, which is very intuitive.

A recursive partitioning analysis was performed on the size of burnt land with all the independent variables. Figure 10 is the tree view of the partitions for the size of burnt land. The figure shows where partitions were created with the mean, standard deviation, and the number of data points for each partition. The first partition was created by the fuel capacity of the delivery aircraft followed by the number of delivery aircraft, and the retardant weight. The right side branch of the first partitioning occurred when the fuel capacity was less than 680 lb. For those 1860 simulation runs, the mean size of burnt land was 3549 ha. When the fuel capacity was more than 680 lb, the mean burnt land was 356 ha. Among those cases where the fuel capacity was more than 682 lb, the number of firefighting aircraft and retardant weight played the key role in successful firefighting. Following the left side of the branches in the figure, the combinations of fuel capacity higher than 682 lbs, more than four firefighting aircraft with a retardant weight of more than 4492 lbs had only 0.69 ha of burnt land with a standard deviation of 0.4 ha. This type of analysis allows us to identify the ranges of MoPs that enable the desired MoE outcomes.

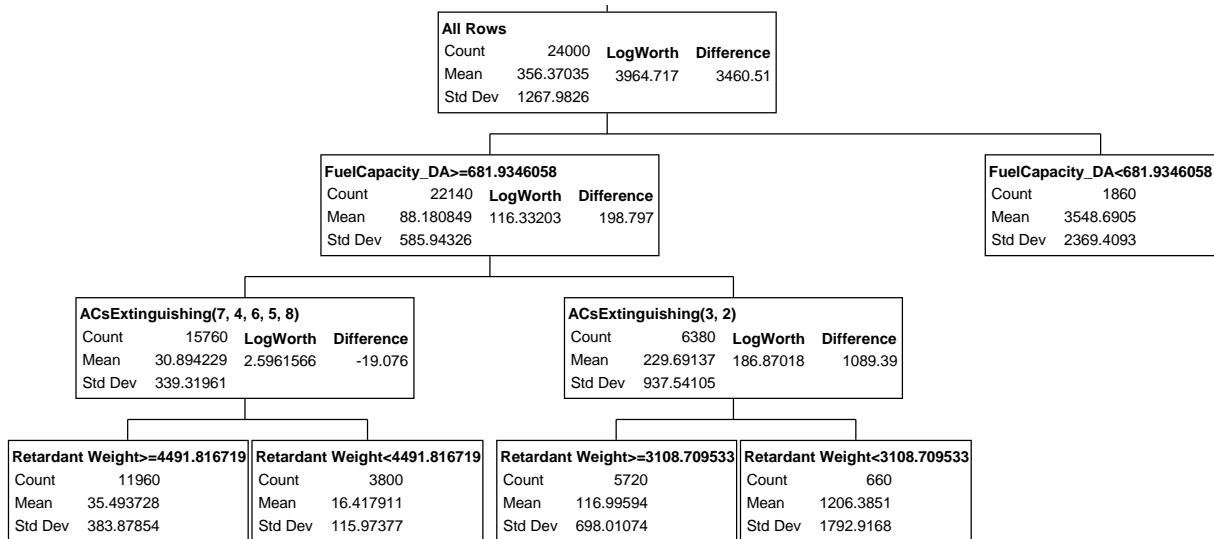


Figure 10. Tree View and the Statistics of the Partitions by the Size of Burnt Land.

3. Investigation of Desirable Aircraft Performance by Inverse Design

While the mean size of the burnt land provides meaningful information in terms of the overall goodness of the family of aerial firefighting fleet, the decision makers might be more interested in the extreme cases than the average cases. For instance, the objective of the customer could be to keep the size of the burnt land less than 5 ha. Or maximize the probability of putting out the fire within 3 hours. These scenarios would be relevant when the customers are interested in acquiring the best performing aerial firefighting fleet. On the other hand, the decision makers might be only interested in preventing catastrophic events with minimum cost. In such a scenario, one would be interested in how far the aircraft performance requirements could be relaxed without losing mission effectiveness. The SoS simulations developed in this paper allow quantitative analyses under various scenarios.

In order to identify the key aircraft performance characteristics attributed to successful firefighting, a constraint on the size of burnt land was applied at 5 ha. The constraint was applied using the scenario filter shown on the left side of Figure 11. Out of 24,000 simulations, only 6942 cases met the imposed constraint. The histograms of the entire dataset and of those cases that met the constraint are compared in Figure 11. Histograms in light green represent the entire 24,000 data points. The histograms in dark green are the simulations where the size of the burnt land was less than 5 ha. For each of the bars of the histograms, the ratio between the dark portion (successful cases) and the light green portion (all cases within the range) is the probability of burnt land being less than 5 ha for the particular range of the given input variable. By observing how the probability changes, the sensitivity of the probability with respect to each variable can be observed.

The histogram shown on the left side of the figure is the distance from the closest airfield to the fire. It shows that the probability of success reduces as the distance is increased. The middle and right side of the figure are the histograms for the search and delivery (firefighting) aircraft. Since all the aircraft performance parameters were input to this SoS simulation and the data samples were generated using a space filling DoE, the histograms for the input variables in light green show uniform distributions as expected. Histograms of the successful cases for some of the input variables such as velocity, TSFC, and L/D still exhibited uniform distributions. It implies that those input variables had little significance in keeping the size of the burnt land less than 5 ha. On the other hand, the dark green histograms of the number of search aircraft, the number of delivery aircraft, and the retardant weight on the delivery aircraft showed gradual increase in their height as the value for input variable increased. This observation indicates that increasing the number of aircraft or the quantity of retardant would have further increased the probability of putting out fire before it grew more than 5 ha. One notable input variable was the fuel weight of the delivery aircraft. The height of the dark green bar increased sharply initially (shown in red circle) and then remained relatively flat after about 1000 lb. The changes in the dark green bars of fuel weight confirm the observations made in the previous sections that it was critical for the delivery aircraft to carry more than certain amount of fuel. Once it had enough fuel, then carrying even more did not improve the probability of success further.

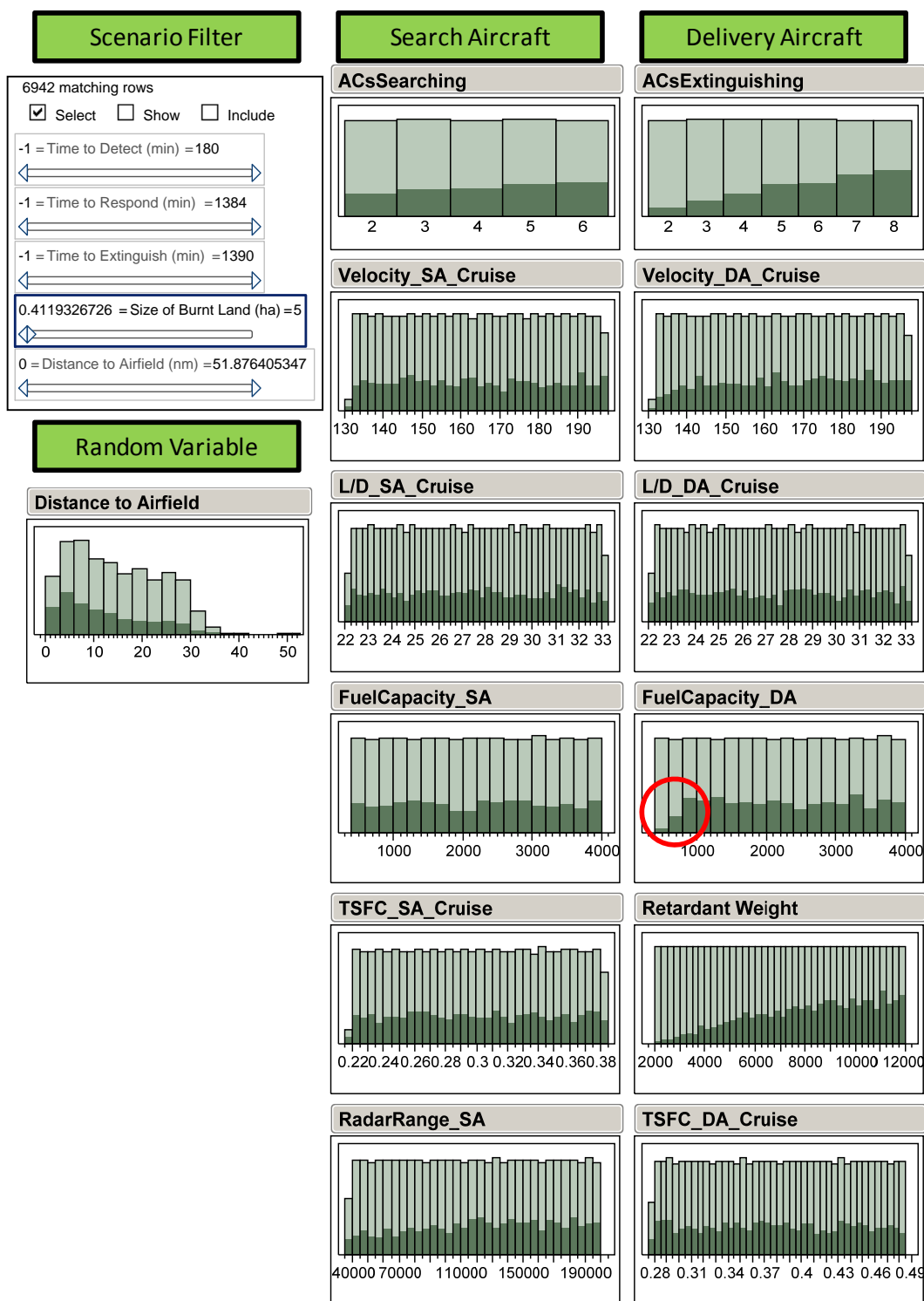


Figure 11. Scenario Filters and Histograms of the Aircraft Characteristics and a Random Variable.

4. Surrogate Modeling

The partitioning shows a very strong bimodal behavior for the burnt land. This result is expected because the airtankers either can extinguish the fire easily or the fire continues to burn until the simulation reaches 24 hours. A neural network was used to capture the nonlinearities and variable interactions within the data. Because of the extreme values in those points that failed to extinguish the fire, models that attempted to include the whole data set

were poor. However, by filtering out the points which failed to extinguish the fire, the resulting neural network had a model validation error R^2 of 0.91. Using the neural network for predicting the median burnt land shows some important interactions between the different variables. Table 2 shows the baseline input values into the neural network.

Table 2. Baseline Case.

ACs Extinguishing	3
ACs Searching	3
Velocity DA Cruise	145
Velocity Sa Cruise	145
L/D DA Cruise	27.38
FuelCapacity DA	2100
Retardant Weight DA	4000
TSFC DA Cruise	0.3897
Radar Range SA	52000

Figure 12 shows the contour plot for the baseline and two perturbed cases with iso-burnt-land lines drawn. The intercept point of both 3 aircraft for search and delivery is marked to allow for better reference in the graphs. In one perturbed case the search aircraft's radar detection range has been doubled, in the other case the delivery aircraft's retardant weight is doubled. In comparing these two cases to the baseline both behave intuitively. Increasing the detection range causes a strong shift in the number of search aircraft needed. Increases in the amount of retardant each delivery aircraft can drop causes a significant decrease in the numbers of both search and drop aircraft required.

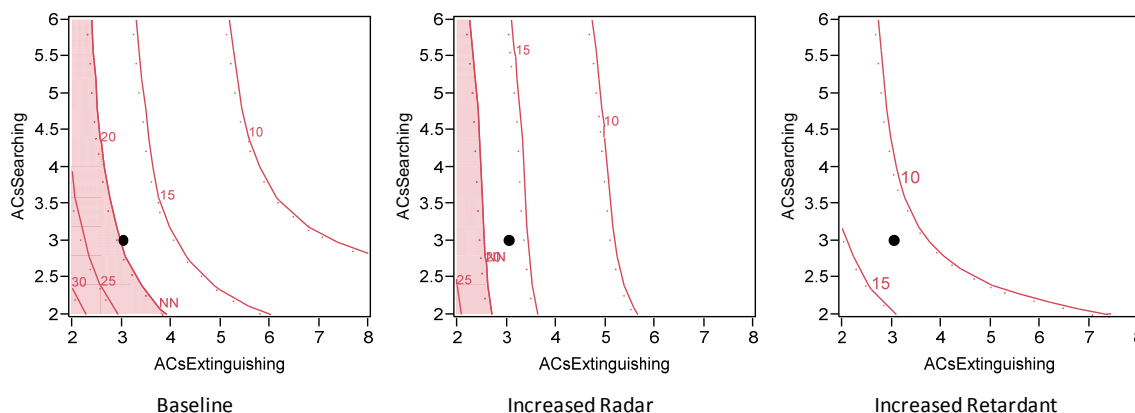


Figure 12. Contour Plot for Burnt Land Neural Network.

Figure 13 shows a prediction profiler for the baseline and perturbed cases. A prediction profiler graphs the partial derivatives in a model and can be used to demonstrate the interactions between the different inputs. For example in the increased radar case, the tail in the number of search aircraft is decreased and the impact from higher search aircraft cruise speeds is smaller. It was expected to have lessened aircraft performance requirements by increasing the radar performance. Similarly, for the higher retardant case, the number of delivery aircraft is decreased and the velocity of the delivery aircraft has less impact.

Neural networks were also used to fit the standard deviation of the burnt land. However there were not enough points to allow the neural networks to converge to a useable model with acceptable error. To be able to fit the standard deviation more repetitions should be included to allow for better convergence.

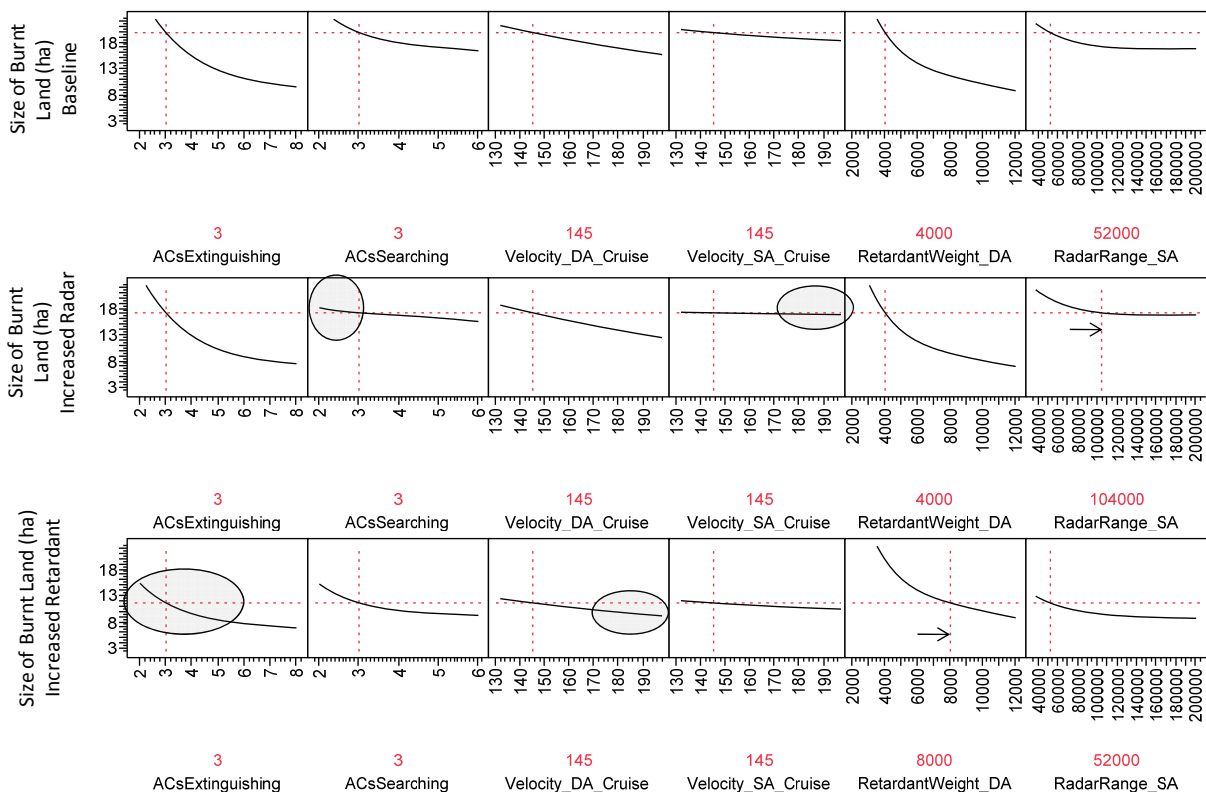


Figure 13. Prediction Profiler for Baseline and perturbed cases.

V. Conclusion

Two agent-based SoS simulations were developed to create a mapping from aircraft performance metrics to mission-level customer requirements. These simulations of a maritime monitoring environment and an island-firefighting environment model civil applications of UAVs. These problems involve missions with conflicting requirements making the design decisions challenging. The agent based simulations created are a step in the preliminary design approach that helps define requirements for the family of UAV systems. Recognizing the challenges running and analyzing stochastic simulations while preserving as much information as possible, various techniques to address such challenges were addressed and demonstrated with the aerial firefighting example. The on-going research effort is aimed at comparing the outputs of both SoS simulations to understand the potential for product family. Another great challenge would be to use these simulation environments to apply all the existing regression techniques suitable for stochastic problems and to compare them. Finally, the error analysis is very challenging in this situation given all the possible sources of uncertainty. It is thus important to be able to verify the surrogate model by comparing it to the simulation and then to validate results with reality.

Acknowledgments

This study was generously supported by Dassault Aviation and Saab Group under fellowships. Philippe Ranque is a former Dassault fellow and Dane Freeman is a Saab fellow through the Aerospace Systems Design Laboratory at the Georgia Institute of Technology. The authors would like to thank Michel Ravachol at Dassault Aviation and Gunnar Holmberg at Saab Group for their guidance and advice.

References

1. Meyer, M.H. and J.M. Utterback, *The product family and the dynamics of core capability*. 1992: International Center for Research on the Management of Technology, Sloan School of Management, Massachusetts Institute of Technology.

2. Ackoff, R.L., *Towards a System of Systems Concepts*. Management Science, 1971. **17**(11): p. 661-671.
3. Ryan, A., *A Multidisciplinary Approach to Complex Systems Design*, in *Applied Mathematics*. 2007, The University of Adelaide.
4. Gorod, A., B. Sauser, and J. Boardman, *System-of-Systems Engineering Management: A Review of Modern History and a Path Forward*. Ieee Systems Journal, 2008. **2**(4): p. 484-499.
5. Sousa-Poza, A., S. Kovacic, and C. Keating, *System of Systems Engineering: an Emerging Multidiscipline*. International Journal of System of Systems Engineering, 2008. **1**(1/2).
6. GPO, U. *Restructuring of the Strategic Defense Initiative (SDI) Program*. 1989.
7. Sage, A.P. and C.D. Cuppan, *On the Systems Engineering and Management of Systems of Systems and Federations of Systems*. Information Knowledge Systems Management, 2001. **2**(4).
8. Kotov, V., *Systems of Systems as Communicating Structures*. Hewlett Packard Computer Systems Laboratory. 1997, Paper HPL-97-124, 1--15.
9. Carlock, P.G. and R.E. Fenton, *System of Systems (SoS) Enterprise Systems Engineering for Information-intensive Organizations*. Systems Engineering, 2001. **4**(4): p. 242-261.
10. Manthorpe, W.H.J., *The Emerging Joint System of Systems: A Systems Engineering Challenge and Opportunity for APL*, in *Johns Hopkins APL Technical Digest*. 1996. p. 305.
11. Maier, M.W. *Architecting Principles for Systems-of-Systems*. [cited 2010; Available from: <http://www.infoed.com/Open/PAPERS/systems.htm>].
12. Boardman, J. and B. Sauser. *System of Systems - the meaning of of*. in *IEEE/SMC International Conference on System of Systems Engineering*. 2006. Los Angeles CA.
13. Meilich, A. *Systems of Systems (SoS) Engineering & Architecture Challenges in a Net Centric Environment*. in *IEEE/SMC International Conference on System of Systems Engineering*. 2006. Los Angeles CA.
14. McInvale, H.D., *Optimal Control Policies for Stochastic Networks with Multiple Decision Makers, in Interdisciplinary Studies: Civil and Environmental Engineering*. 2009, Vanderbilt University: Nashville.
15. Crossley, W.A., *System of Systems: An Introduction of Purdue University Schools of Engineering's Signature Area*. 2004.
16. Agusdinata, D.B. and L. Dittmar. *System-of-Systems Perspective and Exploratory Modeling to Support the Design of Adaptive Policy for Reducing Carbon Emission*. in *System of Systems Engineering, 2007. SoSE '07. IEEE International Conference on*. 2007.
17. Giunta, A.A., *Use of Data Sampling, Surrogate Models, and Numerical Optimization in Engineering Design*. AIAA Paper, 2002. **538**: p. 2002.
18. Laboratory, S.N., *Uncertainty Quantification*, in *DAKOTA 101*. 2009, Lockheed Martin Company.
19. Sacks, J., et al., *Design and Analysis of Computer Experiments*. Statistical Science, 1989. **4**(4).
20. Rosenberger, J. *Crossed Array Design*. Design of Experiments 2010; Available from: <https://onlinecourses.science.psu.edu/stat503/node/75>.
21. Rosenberger, J. *Combined Array Design*. Design of Experiments 2010; Available from: <https://onlinecourses.science.psu.edu/stat503/node/76>.
22. Abramovich, F., T.C. Bailey, and T. Sapatinas, *Wavelet analysis and its statistical applications*. The Statistician, 2000: p. 1-29.
23. Koenker, R. and K.F. Hallock, *Quantile regression*. The Journal of Economic Perspectives, 2001. **15**(4): p. 143-156.
24. Takeuchi, I., et al., *Nonparametric quantile regression*. Journal of Machine Learning Research, 2005. **7**: p. 1001-1032.
25. *Bonn Agreement Aerial Operations Handbook*. 2009, Bonn Agreement Accord de Bonn.
26. Ntamo, L., et al., *Forest fire spread and suppression in DEVS*. Simulation, 2004. **80**(10): p. 479.
27. *Live Ships Map - AIS - Vessel Traffic and Positions [online real-time database]*. Available from: <http://www.marinetraffic.com>.
28. Commission, O., *Geography, hydrography and climate*, in *QSR 2000*. 2000.
29. Wilensky, U., *NetLogo User Manual. Version 4.0*. 2007, Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University.
30. Gaudard, M., P. Ramsey, and M. Stephens, *Interactive Data Mining and Design of Experiments: the JMP® Partition and Custom Design Platforms*. Retrieved April, 2006. **26**: p. 2010.