PARAMETRIC SYSTEM EFFECTIVENESS ANALYSIS USING ARTIFICIAL NEURAL NETWORKS FOR BATTLE MANAGEMENT

Patrick T. Biltgen, Dimitri N. Mavris
Georgia Institute of Technology, Atlanta, GA, USA

Keywords: *Systems-of-Systems, Neural Networks, Battle Management*

Abstract

One of the major challenges facing the simulation community is the extensive reliance upon monolithic simulations that are neither agile nor composable and are inappropriate for capability-based conceptual design. Furthermore, many simulations require extensive use of humans to observe simulated battlefield outcomes and recommend courses of action. This paper summarizes a technique that uses artificial neural networks to provide automated course-of-action selection in a constructive simulation, enabling "exploratory analysis" of a large possibility space without reliance on a human-in-the-loop during analysis runs. The proposed technique is used to understand sensitivities and examine new technologies and tactics in a rapid and traceable manner.

1 Introduction

In the past five years, the US Department of Defense (DoD) has shifted from a tradition threat-based acquisition policy that tended to produce stovepiped individual systems toward a capability-based approach design to foster interoperability and the acquisition of "born joint" capabilities. The official acquisition policy, the Joint Capabilities Integration and Development System (JCIDS) introduces several new analysis steps but provides little guidance for their execution [1]. In 2006, the Joint Chiefs of Staff published recommended guidance on the performance of Capability-Based Assessments (CBAs) which recommends the use of simulation tools where appropriate for Functional Solution Analysis [2]. The use of simulation to address hard military problems is not new; however, simulation has found increasing popularity for the conceptual design of complex systems and is recommended by both the US National Science Foundation (NSF) and the US National Research Council (NRC) for use in the systems engineering process [3], [4].

The impetus to elevate systems analysis to the "systems-of-systems" level to assess the effectiveness of military capabilities introduces a number of challenges into the conceptual design of complex systems. First, in addition to modeling the complex phenomenologies of current and future systems and their technologies, analysts and designers must now place the system in an appropriate operational context, define scenario parameters, and understand the interoperation of a candidate system with other elements of a system-of-systems. This often greatly increases the burden on simulation systems and often requires collaboration across multiple partners for model development, integration, and code execution. Second, many simulation systems were designed not for large exploratory analysis of many combinations, but rather, were written to perform validated analysis of a few point scenarios [5]. Many such tools use a human as part of the analysis process to assess the simulated battlefield at predefined checkpoints and recommend courses of action. Third, designers now operate in unfamiliar design spaces that exceed their sense of intuition. At the system-of-systems level, factors that most

often appear to be drivers may be dominated by the design variables of other systems, technologies, the environment, or even the methods of employing those systems.

This paper addresses a methodology for automating the selection of military tactics within a constructive simulation to remove the human from the execution loop, incorporating physics-based models of systems and technologies, and using simulation-based methods to explore unfamiliar regions of the design space to provide insight to engineers and decision makers. A more detailed treatment of this methodology is outlined in References 6 and 7.

2 Using Simulation for Capability-Based Design

The first major challenge outlined above, the need to represent phenomenologies and put the system in a relevant operational context, is addressed through the use of simulation.

Simulation, the imitation of physical phenomena with mathematical formulas, "provides a powerful alternative to the techniques of experimental science and observation when phenomena are not observable or when measurements are impractical or too expensive" [8], [3]. Simulation has long been an invaluable tool in military planning. Advances in computing power and proliferation of affordable simulation software have widened the user base for simulations as a tool for exploratory analysis of potential concepts early in the design process.

Simulation of systems-of-systems is best enabled through the use of a hierarchical object-oriented simulation environment [9]. In contrast to monolithic hardcoded simulation systems that were purpose-built to examine the interplay of certain systems or scenarios, extensible simulation frameworks allow multi-fidelity, multi-resolution modeling of a number of systems and their integrated effects [10]. A popular framework (used herein for a proof of concept) is the FLAMES framework by Ternion, Inc [11]. Based on object-oriented C, FLAMES allows the creation of flexible simulations using a number of user defined objects and instantiations of those objects. Recent versions have included the capacity to define common interfaces, facilitating model sharing and interoperability. FLAMES also supports execution of design-of-experiments or optimization through external interfaces. The key benefit of environment such as FLAMES is the ability to assemble a simulation environment that encompasses many different classes of military equipment and their associated "behaviors," and explore the interaction between the system design variables and a common set of Measures of Effectiveness (MoEs) that quantify outcomes in capability terms.

Flexible simulation environments should allow system design variables, scenario parameters, friendly and enemy locations, and specific courses of action (behaviors or tactics) to be altered to observe the impact on one or more MoEs. The ability to incorporate variable fidelity models allows the simulation environment's run-time to be scaled based on the fidelity needs of the simulationist. Varying only those system properties determined to be most significant to the overall variation of the MoEs is also an efficient way to perform large scale studies.

3 Battle Management in Simulation

Battle Management is defined by DoD as "The management of activities within the operational environment based on the commands, direction, and guidance given by appropriate authority" [12]. The battle manager is therefore, the individual who provides commands, direction, and guidance based on an observed situation. In the most basic sense, the battle manager is a resource allocator. He or she must select which resource to assign to which task, and when… knowing that a higher priority task requiring the allocated asset may emerge at any time. Resource allocation decisions are therefore dependent on the specific situation within a given scenario including geography, time, asset locations, weapon loadouts, and the dispersion of hostile threats across the battlespace. In military operations, battle

management is handled by local or combatant commanders who assign courses of action to forces under their command. In many simulation systems, this role is simulated by one or more trained experts who interface with a simulation system. Most simulation activities use visualization facilities to depict the state of the battlespace at predefined checkpoints (or after certain events are reached) and present the battle manager with several courses of action from which to choose. The orders and intent are sent to the simulated entities and the computer simulation resumes until the next checkpoint. Though they excel in terms of fidelity and realism, these simulation activities may require hours or days for a single run. For this reason, traditional human-based battle management approaches are inappropriate for conceptual capability-based design.

The second challenge outlined above is how to reduce the dependence on human operators to enable large-scale probabilistic studies of many possible alternatives. One popular method for battle management in constructive simulation is the use of decision trees. Decision trees consist of predetermined courses of action that are executed either with a specified probability or upon the attainment of certain operational conditions. Decision trees (or pre-scripted battle management logic) are an appropriate technique when the courses of action are known and can be related to the set of resources available. Unfortunately, the third challenge outlined in the introduction comes into play here: when studying future systems, it may not be possible to define the set of rules for their use. Just as the physics of new technologies are often unknown early in the conceptual design process, the ideal concepts for employment for these systems may also be difficult to determine. This is because the experts from which an experience base of tactics and doctrine are drawn have been exposed to a set of known situations and formulate courses of action based on this experience. A technique for computer assisted battle management that develops appropriate courses of action is needed.

4 Agent-Based Modeling and Machine Learning

Complex systems are categorized by emergent, dynamic, non-linear behavior derived from interactions between lower level components. The field of agent-based modeling and simulation (ABM/S) uses a bottom up approach to the design of complex systems that relies on creating relatively simple “agents” and defining the interactions between agents in such a way to generate realistic system-level behavior with relatively unsophisticated subsystem elements. According to Ilachinski, “agent-based simulations of complex adaptive systems are predicated on the idea that the global behavior of a complex system derives entirely from the low-level interactions among its constituent agents” [13].

Machine Learning, a subdiscipline of artificial intelligence, is a process by which a predictive computer improves its performance over time. The use of machine learning techniques such as genetic algorithms or recursive neural networks to develop adaptive intelligent agents has received much attention over the past fifteen years and has been applied to a number of disciplines from speech recognition to video games.

Several techniques are used to develop intelligent agents. Analytic Learning systems rely on a large knowledge base of problem classes and compare presented problems to the most similar solution. In Inductive Learning, an external “teacher” is used to provide problem examples and grade the program’s performance at addressing them to provide increased understanding over time. Connectionist Learning uses a set of exemplars and a mathematical algorithm to develop patterns that closely match the provided data. This approach most often uses artificial neural networks to formulate the patterns. Finally, Selectionist Learning “evolves” toward the correct answer using a trial-and-error approach where exemplars are tested over time. In practice, a genetic algorithm is traditionally used for this approach [14].

While a continuously learning algorithm is most appropriate to simulate the generic battle management problem, if the intent is to compare conceptual system solutions to one another, an algorithm with a consistent level of training is required. When continuously learning algorithms are used for conceptual design, the learning rate of the algorithm confounds concept comparison: a "dumb" algorithm with "good" technologies may underperform a "smart" algorithm with "poor" technologies.

The Connectionist Learning technique can be used to develop an algorithmic battle manager that is capable of developing detailed courses of action when only general ones exist. This method is appropriate for automated battle management of constructive simulations used for concept comparison and technology evaluation [6].

5 Neural Network Algorithms for Battle Management

The concept of artificial neural networks (ANNs) was first introduced by neuro-physiologist Warren McCulloch and mathematician Walter Pitts in 1943 and was developed extensively in the 1990s [15]. In contrast to metamodeling techniques that assume a functional form of the response (response surface equations or Kriging models), artificial neural networks use a generic functional form and rely on a series of unknown coefficients which are "tuned" through an iterative process called training the neural network. The power of the technique comes from the use of a sigmoid equation as an activation function from the set of inputs, *X*, to the set of outputs, *Y* through a user-defined number of "hidden nodes," *H*. A commonly used construct for ANNs, the multi-layer perceptron is shown in Fig. 1 [16]. This construct is trained using a process called backpropagation, where an estimated output pattern is compared with a known output pattern and the unknown coefficients are tuned through an optimization algorithm until the prediction error between the actual and predicted values is minimized.

There are many different architectures for neural networks including radial basis functions, recurrent neural networks, self-organizing maps, Hopfield networks, associative neural networks, and others [17], [18]. The architecture used in this work is the feedforward neural network shown in Fig. 1.

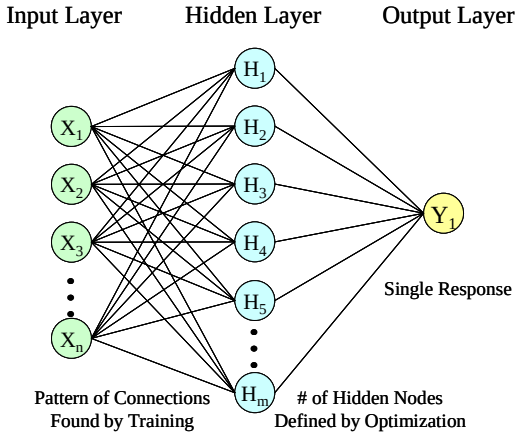

Fig. 1. Structure of a Feedforward Neural Network.

Since the objective of training a neural network for battle management is to create a surrogate of a human decision maker, the neural network can be thought of as a "Meta-General" that is capable of understanding battlefield situations and allocating resources based on an observed situation. This paradigm is illustrated in Fig. 2.

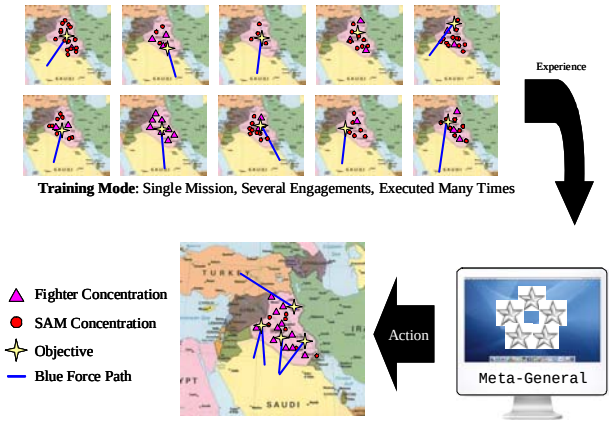

Fig. 2. Process for Creating a "Meta-General" [6].

In this construct, a number of training situations (the aforementioned exemplars) are provided by generating a friendly and enemy

order-of-battle and assigning properties to the blue assets that may be allocated. A number of iterations are performed to build an experience base of exemplars. The neural network is then trained using the input/output data from these simulation executions as the patterns for learning. The resultant neural network (a mathematical equation with defined coefficients) is inserted into the simulation code. The input variables describe the battlefield situation and the status of available resources. The responses are related to the courses of action that may be undertaken by the battle manager. The specific variables used and the reasoning rationale are explained in a subsequent section.

When faced with a new situation, the trained battle manager operating in Analysis Mode as shown in Fig. 2, identifies the situation in terms of geography and threat laydown and iterates through available resources to develop the optimal asset-to-target pairings based on the "knowledge" encapsulated in the neural network.

6 Procedure for Training the Meta-General

To use this concept for battle management, a number of enablers are required. First, a simulation must be constructed that contains physics models of the equipment to be tested or compared. Second, a scenario must be defined that describes the battlespace on which the equipment is expected to operate. Third, general instructions related to missions and outcomes must be defined in computational terms. In this example problem, physics models describing aircraft and weapon flight were developed. The scenario defined a geographic area, a set of targets, a threat laydown, and a priority of generic target classes. The mission of the aircraft in the simulation was to attack the targets in prioritized order (if possible). The purpose of the battle manager is to decide which assets are best suited to attack each target at a given time. This decision is constrained by which aircraft are currently available and the state of enemy air defenses. A generic, unclassified scenario was constructed using the FLAMES framework and was based on public domain data related to the 1991 Persian Gulf War, Operation Desert Storm [18-22]. The scenario used is depicted in Fig. 3.

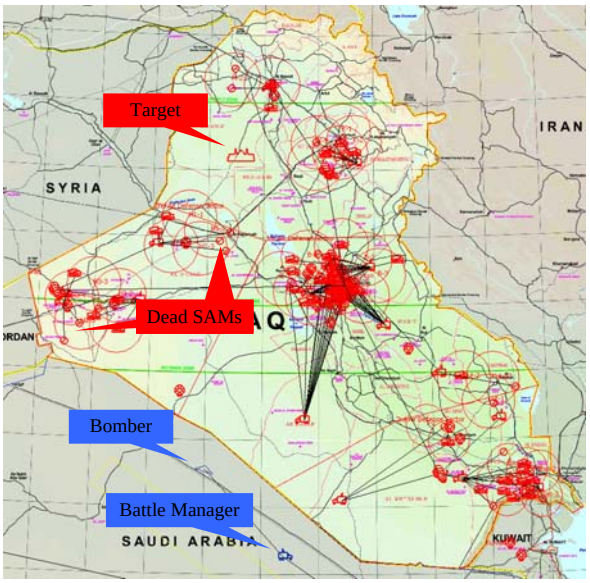

Fig. 3. Scenario Used for Battle Manager Training [6].

Next, the ranges of the parameters used for the battle manager's inputs are defined based on the environment and parameters describing the friendly systems. The variables used are shown in Tab. I.

Tab. I. Input Parameters for Meta-General Training.

Variable	Low	High
Max Platform Speed (Mach)	0.72	4
Cruise Altitude (ft)	10,000	50,000
GTOW (lbs)	35,000	1,200,000
Empty Weight Ratio	0.4	0.55
Payload Weight (lbs)	2,000	80,000
Thrust/Weight	0.35	1.5
Wing Loading (lb/ft^2)	20	150
Drag Coefficient	0.01	0.09
Maximum Lift Coefficient	1.5	3
TSFC (lb_m/lb_f-hr)	0.3	0.8
Susceptibility Factor	0.01	1
Weapon Range (nm)	10	1,200
Weapon Speed (Mach)	0.72	6
Bomber Location (Lat)	14°	37°
Bomber Location (Lon)	34°	60°
Target Location (Lat)	29°	37°
Target Location (Lon)	38°	48°
SAM Density (%)	0	100

These parameters were determined from a larger list of aircraft and weapon parameters by screening for those parameters with the greatest impact on the overall outcome of the simulated conflict. The first ten input variables are design variables for a fixed wing aircraft that are generally under a designer's control. *Susceptibility Factor* is a surrogate for the aircraft's "stealth" and essentially acts as a probability of detection by hostile forces. Larger values increase the likelihood that the friendly aircraft is detected by the adversary's air defense system. The next two variables define the properties of air-to-ground weapons and allow a range of weapons from glide bombs to cruise missiles to be simulated.

The next five parameters define the geometric and threat situation of the battlespace. The *Bomber Location* defines the current location of the friendly bomber and the *Target Location* defines the geographic location of the target under consideration. The final parameter, *SAM Density* describes the current intensity of surface-to-air missile (SAM) defenses in terms of the percentage of sites still alive. These parameters are used to determine whether a friendly bomber will fly over SAM defenses while enroute to the target under consideration and whether the sites are alive. If the *SAM Density* is 100%, it is highly likely the friendly platform will be attacked if the *Susceptibility Factor* is non-zero. On the other hand, when the *SAM Density* is low, this implies that most of the defenses have been destroyed by other assets and most aircraft will be able to prosecute the war with impunity.

To generate data for neural network training, the Design-of-Experiments (DOE) technique is used. DOE is "a systematic, rigorous approach to engineering problem solving that applies principles and techniques at the data collection stage so as to ensure the generation of valid, defensible, and supportable engineering conclusions" [24]. Since the 1920s, a number of techniques have been developed for analysis of critical factors and the generation of response surfaces. Since ANNs require dense population of the interior of the design space to develop accurate estimates of nonlinear behaviors, traditional DOE methods such as fraction factorial or central composite designs are not appropriate. Space-filling experimental designs are the most effective class of designs for generating ANNs. Of these, there are two general properties:

- Maximize the separation of sampled points.
- Maximize the dispersion across the sample space.

While random points can be used to fill a space, an alternative scheme called "sphere-packing" is used to minimize the maximum distance between any two points in *n*-dimensional space, akin to placing billiard balls into an *n*-dimensional box. This tends to ensure maximum separation of the points in the experimental design and, in general, improves the fit of resultant surrogate models. Mathematical techniques to assess this distance have been developed extensively in the literature [25].

In addition to the maximum spacing criteria, another useful criterion for space-filling designs is the property of uniform spacing across the region of interest. According to Cioppa, "A good space-filling design is one in which the design points are scattered throughout the experimental region with minimal unsampled regions; that is, the voided regions are relatively small" [26]. A uniform design seeks to spread the points evenly by minimizing a parameter called the "discrepancy," which is the difference from the empirically sampled set of design points and a theoretical uniform distribution. This type of design directly addresses the second property above.

The final type of space-filling design discussed in this work is the Latin Hypercube (LHC) designs, developed by Ronald L. Iman, J. C. Helton, and J. E. Campbell in the early 1980s. A LHC attempts to distribute points evenly through the design space using a combination of uniform designs with a sphere-packing scheme.

Upon experimental comparison of all three DOE types and a set of random points, DOEs created with the sphere-packing scheme tended to produce the most accurate ANNs for this application [6].

An eighteen variable 10,000 case space-filling design was developed using the JMP® 7.0 software package and executed through the FLAMES-based simulation depicted in Fig. 3. Of the 10,000 cases, 8,922 were "usable" due to the fact that the square boundaries of the DOE input ranges occasionally placed the target outside the allowable bounds of the simulation (in a country other than Iraq), resulting in a failed case.

Two predictive neural network equations were developed to answer two questions facing the battle manager:

"If aircraft/weapon combination (*i*) is tasked to attack target (*j*)"

- Will platform *i* be lost? *(Platform Lost)*
- Will target *j* be killed? *(Target Killed)*

Note that in this formulation, the probabilities of each event are not considered, only the deterministic outcome of the engagement. Therefore, the neural network predicts the response in the range [0,1].

Many computerized tools exist to automate the neural network training process, which is in practice a time consuming iterative effort to tune the unknown coefficients of the equation to minimize the error between the actual and predicted values resulting from the 8,922 DOE runs. A computational routine written by Johnson and Schutte around the MATLAB Neural Network Toolbox was used to perform the neural network training [27]. The resultant eight hidden-node ANN for *Platform Lost* has an overall accuracy of 96.54%. A nine-node ANN for *Target Killed* has an overall accuracy of 97.44%. Here, the accuracy statistics represent the probability that the battle manager makes the correct decision given the information provided about the state of the simulation is within the original ranges of the DOE table (Tab. I).

7 Using the Meta-General in a Simulation

The aforementioned section described how to set up the neural network architecture, define the DOE parameters, and train the neural network using the MATLAB neural network toolbox. The result of the previous steps are two equations that each have eighteen input variables and produce a single Boolean response. The intent of this equation is to give a computational routine predictive power on the expected outcome of a potential engagement pairing.

This neural network equation is inserted into simulation code which receives a prioritized target list and iterates through all unassigned resources (platforms). Each platform carries one or more weapons. At run-time, the battle manager code queries the current platform and each weapon for their properties (13 of the 18 required inputs), queries the simulation kernel for the location of the platform and target (4 of the 18 required inputs), and counts the percentage of SAM sites currently alive[1].

The battle manager uses the outputs of the neural network equation to evaluate a "cost" for each potential engagement using the formula:

$$\begin{aligned}\text{EngagementCost} &= (T_K)\text{TargetCost} - (P_L)\text{BomberCost} \\ &- (Time)\text{TimeCost} - \text{MunitionCost}\end{aligned} \quad (1)$$

Where T_K and P_L are the Boolean outputs of the neural networks for *Target Killed* and *Platform Lost* respectively. *TargetCost* is a dollar amount assigned to represent the value of a target. *BomberCost* is weight-based regression of aircraft cost based on empty weight. *MunitionCost* is a similar regression to represent the relative cost of different munitions. *Time* and *TimeCost* take into account, all other factors being equal, that the battle manager would prefer to use a closer asset to an identical asset with a slower response time. Using this equation, the battle manager code iterates through all possible resources and identifies *the pairing with the highest positive cost*. In this manner, the battle manager will not allocate a bomber to a "suicide mission" when the cost of the platform exceeds the value of the target.

[1] Note that in reality, the calculation of *SAM Density* is impractical. One of the primary difficulties in combat is determining the location and status of enemy defenses. For this proof of concept exercise, the confounding factors of surveillance and intelligence are abstracted out using this simplification.

Similarly, the battle manager will not allocate a munition whose cost is more than the user-specified value of the target.

The code iterates until either all targets have been eliminated or until there are no targets that can be prosecuted with the platform/ weapon combinations available to the battle manager. More information on this procedure is available in References 6 and 7.

8 Observing the Meta-General in Action

When using the cost prioritization method shown in Equation 1, intuitive patterns may be observed in the simulation. Early in the simulated conflict when *SAM Density* is high, only the stealthiest platforms are likely to survive. The Meta-General tends to allocate either stealthy platforms with short range weapons or unstealthy platforms with long range weapons. Recall that the logic in Equation 1 prevents the battle manager from making decisions that intentionally sacrifice platforms. Additionally, the Meta-General tends to prefer less expensive weapons provided that the platform will not be lost.

In another test, the battle manager was asked to route the assets along a preselected set of twenty different airspace corridors with varying length and threat levels. The threat level, or amount of danger associated with each corridor is calculated by how many SAM sites' range circles intersect each corridor. A plot of corridor usage over time is shown in Fig. 4. Early in the conflict, the battle manager tends to prioritize usage of the "Long, Very Safe" corridor for all assets regardless of their level of technology. As time passes and some SAM sites are destroyed, the *SAM Density* parameter is reduced which provides other avenues for attack for the friendly aircraft. Shorter, more dangerous routes are preferentially selected later in the conflict. Even though they are classified as dangerous, these routes are often selected because the SAM sites formerly protecting them have been destroyed by actions earlier in the conflict. This test demonstrates how the battle manager's decisions change over time to reflect the state of the simulation.

Late in the conflict, the Meta-General's route choices are nearly evenly distributed across the spectrum of possible choices because many of the defenses have been destroyed and even unstealthy platforms with short range munitions can survive in the permissive threat environment.

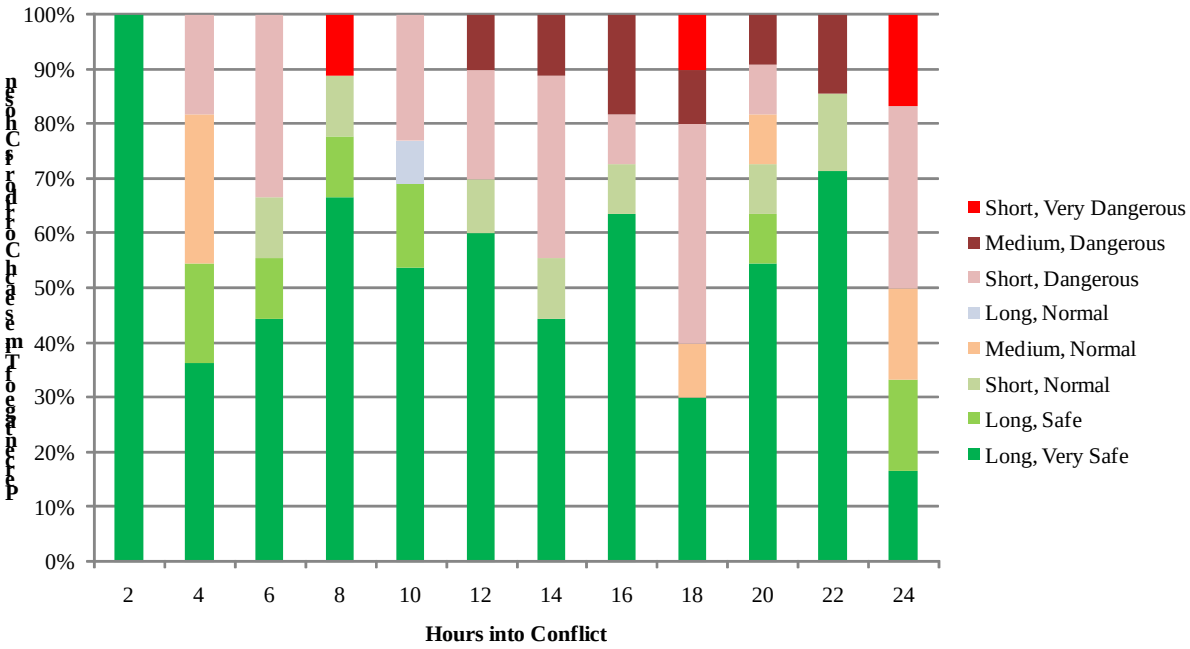

Fig. 4. Corridors Selected by the Meta-General Over Time.

9 Summary and Conclusions

Systems engineers have long been familiar with designing systems to meet requirements; however, with the recent shift to capability-based acquisition, crisp and detailed requirements statements are quickly becoming a thing of the past. Engineers are increasingly asked to examine "multiple combinations of ways and means to achieve an effect" [1].

Computer simulations are a valuable tool for understanding complex relationships endemic to systems-of-systems problems; however, many military simulations rely heavily on hardcoded assumptions or humans with tacit information that may not extrapolate to the systems, technologies, and tactics of the future.

By constructing a synthetic combat environment that allows many potential combinations of systems and courses of action to be quickly prototyped, an experience base can be efficiently generated using DOE methods for regression into an artificial neural network. As demonstrated in this work, an accurate neural network (validated to match the training data) can be used inside of a computer simulation to estimate the "best" potential courses of action for a given suite of aircraft platforms and weapons in a time-variant threat environment.

The combination of agent-based models with a hierarchical object-oriented constructive simulation environment enables parametric system effectiveness analysis for large-scale systems-of-systems by providing a mechanism to evaluate new systems and technologies with respect to capability-based measures of effectiveness.

One of the major drawbacks of the proposed technique is the need to re-train the neural network when either the assumptions behind the scenario or the ranges of the input variables are changed. Scenario assumptions are often changed frequently during the definition phases of simulation activities, and the utility of the proposed Meta-General concept relies on the simulation team composing assumptions and the simulation environment in a structured manner that reduces unnecessary iteration.

By using this technique, technologists can provide a common rationale behind the "behaviors" of assets in the simulation: often the most difficult thing for engineers and technology experts to explain. Since technology and tactics must often be co-developed, the proposed approach shows one way to ensure that the best set of tactics and courses of action are matched to the systems employed.

Agent-based models hold promise for the simulation of human behaviors and the understanding of the complex dynamics endemic to systems-of-systems problems; however, as Lazaroff and Snowden note, they are extremely difficult to validate and are more useful for providing insight into complex phenomena than precise quantification of effects [28]. Nevertheless, the set of modeling tools outlined in this work is becoming increasingly popular in the conceptual design of complex systems.

Future research efforts will apply this set of methodologies to other problems in an effort to develop more precise guidelines for their use and to develop recommendations used to standardize the application of neural networks, constructive simulation, and agent-based models in the systems engineering and exploratory analysis process.

Acknowledgements

The authors would like to thank Mr. David Brown and the U.S. Air Force Research Laboratory for their sponsorship of this research program under Cooperative Agreement FA8650-04-3-9015. Graduate researchers Shuo-Ju Chou, William Engler and Steven Tangen also contributed significantly to this work.

References

[1] United States Joint Chiefs of Staff, "Joint Capabilities Integration and Development System (JCIDS)," CJCSI 3170.01F, May 2007.

[2] United States Joint Chiefs of Staff, *Capabilities-Based Assessment (CBA) User's Guide, Version 2*, Force Structure, Resources, and Assessments Directorate (JCS J-8), December 2006.

[3] "Simulation-Based Engineering Science: Revolutionizing Engineering Science through Simulation," Document Number SBES0506, National Science Foundation, May 26, 2006.

[4] "Pre-Milestone A and Early-Phase Systems Engineering: A Retrospective Review and Benefits for Future Air Force Acquisition," National Research Council, URL: http://www.nap.edu/catalog/12065.html, 2008, [cited May 21, 2008].

[5] Davis, Paul K., Bigelow, James H., and McEver, Jimmie, "Exploratory Analysis and a Case History of Multiresolution, Multiperspective Modeling," RAND Corporation, RP-925, Reprinted from Proceedings of the 2000 Winter Simulation Conference, Jeffrey A. Joines, Russell R. Barton, K. Kang, and Paul A. Fishwick (editors), December, 2000 and Proceedings of the SPIE, Vol. 4026, 2000.

[6] Biltgen, P. T., *A Methodology for Capability-Based Technology Evaluation for Systems-of-Systems*, PhD Thesis, School of Aerospace Engineering, Georgia Institute of Technology, May 2007.

[7] Mavris, D. N., and Biltgen, P. T., "Theory and Implementation of an Agent-Based Intelligent Battle Manager for Quantitative Technology Assessment for Long Range Strike," Cooperative Agreement FA8650-04-3-9015, Final Report to the Wright Brothers Institute and the United States Air Force Research Laboratory, January 22, 2007.

[8] "Definition: Simulation." URL: http://www.webopedia.com/TERM/S/simulation.html, [cited June 1, 2008].

[9] Biltgen P. T., and Mavris, D. N., "A Methodology for Capability-Focused Technology Evaluation of Systems-of-Systems," AIAA-2007-1331, Presented at the 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, Jan. 8-11, 2007.

[10] Davis, Paul K., Bigelow, James, H., "Experiments in Multiresolution Modeling (MRM)," RAND Corporation, MR-1004-DARPA, 1998.

[11] Ternion Corporation, "FLAMES Construction Simulation Framework, Version 6.1," Computer Program, Ternion Corporation, Huntsville, AL, 1991-2006.

[12] U.S. Department of Defense, "DoD Dictionary," Defense Technical Information Center, Joint Publication 1-02, Updated March 4, 2008, URL: http://www.dtic.mil/doctrine/jel/doddict, [cited May 27, 2008].

[13] Ilachinski, A., "Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Warfare (U)," in CRM 97-61.1, Center for Naval Analyses, August 1997.

[14] Ilachinski, A., *Artificial War: Multiagent-Based Simulation of Combat*, World Scientific, 2004.

[15] McCulloch, W. S. and Pitts, W. H., "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[16] Minsky, M. A. and Papert, S. A. *Perceptrons: an Introduction to Computational Geometry*, MIT Press, 1969.

[17] Bar-Yam, Y., *Dynamics of Complex Systems (Studies in Nonlinearity)*, Westview Press, 2003.

[18] Hassoun, M. H., *Fundamentals of Artificial Neural Networks*, MIT Press, 1995.

[19] Davis, R. G., *Decisive Force: Strategic Bombing in the Gulf War*, Air Force Historical Studies Office, GPO Stock No.008-070-00710-0, 1996.

[20] United States Department of Defense, "Final Report to Congress: Conduct of the Persian Gulf War," tech. rep., April 1992.

[21] Cohen, E. A., "Gulf War Air Power Survey, Volume I: Planning and Command and Control," tech. rep., Washington, D. C., 1993.

[22] Putney, D. T., *Airpower Advantage: Planning the Gulf War Air Campaign 1989-1991*, Air Force History and Museums Program, United States Air Force, 1994.

[23] Glosson, B., *War With Iraq: Critical Lessons*, Carolina Gardener, March 2003.

[24] NIST/SEMATECH, "e-Handbook of Statistical Methods." URL: http://www.itl.nist.gov/div898/handbook/, [cited April 1, 2008].

[25] Johnson, M., Moore, L., and Ylvisaker, D., "Minimax and Maximin Distance Designs," *Journal of Statistical Planning and Inference*, vol. 26, pp. 131–148, 1990.

[26] Cioppa, T. M., *Efficient Nearly Orthogonal and Space-Filling Experimental Designs for High-Dimensional Complex Models*, PhD thesis, Naval Postgraduate School, Monterey, California, 2002.

[27] Johnson, C., and Schutte, J., *Basic Regression Analysis for Integrated Neural Networks (BRAINN) Documentation*, Version 1.2. Georgia Institute of Technology, 225 North Avenue, Atlanta, GA 30332, June 14, 2005.

[28] Lazaroff, M. & Snowden, D. "Anticipatory models for Counter Terrorism" in Popp, R & Yen, J, *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*, Wiley-IEEE Press 2006.

Copyright Statement