Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

TRADESPACE ANALYSIS FOR MULTIPLE PERFORMANCE MEASURES

Alex D. MacCalman

Department of Systems Engineering
United States Military Academy
752 Mahan Hall
West Point, NY 10996, USA

Susan M. Sanchez
Mary L. McDonald

Department of Operations Research
Naval Postgraduate School
1411 Cunningham Road
Monterey, CA 93943, USA

Simon R. Goerger

U.S. Army Corps of Engineers
Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180, USA

Andrew T. Karl

Adsurgo, LLC
3700 Quebec St
Unit 100 Suite 258
Denver, CO 80207, USA

ABSTRACT

To meet the changing demands of operational environments, future Department of Defense solutions require the engineering of resilient systems. Scientists, engineers and analysts rely on modeling, simulation, and tradespace analysis to design future resilient systems. During conceptual system design, high performance computing clusters and models from multiple domains are leveraged to conduct large-scale simulation experiments that generate multi-dimensional data for tradespace exploration. Despite recent breakthroughs in computation capabilities, the world's most powerful computers cannot effectively explore a high-dimensional tradespace using a brute-force approach. This paper outlines a viable methodology and process to generate large numbers of variant solutions for tradeoff analysis. Design of experiments is used to efficiently explore a high-dimensional tradespace and identify system design drivers. These drivers are used to identify model inputs that help focus tradespace generation in areas that promise viable solutions. A dashboard illustrates how viable variant exploration can be conducted to illuminate trade decisions.

1 INTRODUCTION

A system design involves hundreds of tradable variables that must be balanced in order to develop a viable system solution that meets the demands of the stakeholders and performs effectively. Understanding the key tradable variables that have the most influence on a system design problem is critical during the conceptual design of a system. A tradeoff is a compromise between objectives such that improving one requires that we degrade another. Tradeoff decisions are based on data, information, and knowledge acquired from simulation model outputs, developmental and operational testing, subject matter expertise, and legacy system architectures. In this paper, we focus on how we can illuminate tradeoffs among multiple simulation outputs by leveraging the methods of experimental design and a dynamic dashboard.

During the system life cycle, in order to evaluate system effectiveness in an operational context, we use a variety of models and simulations that represent different domains; these domains include operational effectiveness, physical feasibility, life cycle costing, manufacturability, reliability, and many more. The inputs to these models and simulations represent the system properties that define the alternative configurations. We are most interested in identifying the system design drivers that have the highest impact on the model output. Currently, there is a technical gap with regard to our ability to identify and understand the system design drivers when there is a high volume of multi-dimensional data. The general state-of-practice for those in the Department of Defense (DoD) system design community who use simulation is to conduct a baseline run and a small set of excursions, often chosen by subject matter experts—but these do not effectively explore the system alternative design space. Large amounts of time, money, and resources are devoted to building complicated simulation models, and we do not use them to the maximum extent possible if we only compare a few excursions from the baseline. The most effective way to determine the system design drivers is to leverage the methods of statistical experimental design—by using design and analysis methods developed explicitly for large-scale simulation experiments (see, e.g., Sanchez 2015; Kleijnen 2015; Sanchez, Sanchez, and Wan 2014; Sanchez and Wan 2015). The field of design of experiments (DOE) allows the analyst to identify which model inputs affect the outputs of interest. DOE provides a number of benefits that can assist in the design of a system. These can include clearly identifying the model inputs that affect the output responses, identifying interactions that may exist between model inputs, uncovering detailed insight into the model's behavior, revealing consequences of certain modeling assumptions, framing questions to ask of the simulation model, challenging or confirming intuition about the relative importance (and direction) of model input effects, discovering which alternatives are robust to uncertainties, and uncovering problems with simulation program logic (Kleijnen et al. 2005).

2 APPROACH

2.1 Statistical Metamodels and the Dashboard

In order to identify and understand the system design drivers across several different domain simulation models, we propose using statistical metamodeling to approximate the simulations' behavior. A statistical metamodel is an empirical model, developed from either observational or experimental design data, that relates a set of inputs to an output (Grayson and Gardner 2015; Hastie et al. 2009). Metamodels can be built using a number of statistical methods that include stepwise regression, boosted trees, neural nets, and bootstrap forests (Grayson and Gardner 2015; Hastie et al. 2009; Kuhn and Johnson 2013). Regression metamodels can provide useful insights about the nature of the simulation model behavior (see, e.g., Barton 2015), while the other metamodeling methods may be better able to predict effectiveness when interpolating between design points without imposing a parametric relationship.

To generate the data needed to fit metamodels, we advocate using space-filling designs because they allow construction of a variety of types of metamodels after the experiment is complete. Space-filling designs allow us to determine the driving factors, detect interactions between input variables, identify "knees in the curve" corresponding to increasing or diminishing rates of change, and find thresholds or change points in localized areas. In this paper, we generate the design using the genetic algorithm of MacCalman, Vieira, and Lucas (2016), enforcing low correlation among all main effects. These designs are efficient, have very low correlations between columns, can handle continuous, discrete, and categorical data, and effectively explore the interior of the experimental design region. Their genetic algorithm can also be used to generate designs that have low correlation among second-order terms as well (i.e., quadratic effects and two-way interactions) if the number of factors is relatively small.

We create a dynamic dashboard using the collection of metamodels to help visualize the multi-dimensional model output landscape using horizontal and vertical cross sections. These cross sections allow us to clearly identify the tradable variables and find viable system variants that met the desired

capabilities across multiple viewpoints and are physically feasible. We can easily visualize the model output landscape with a surface plot when there are only three dimensions. Because there are often several more dimensions in a systems design problem, we developed a dashboard that visualizes horizontal and vertical cross sections of the multi-dimensional model output landscape. We use a contour profiler that is a two dimensional projection that shows a horizontal cross section of a model output landscape within the experimental design region (SAS Institute 2015). Visualizing the selected projections allows the user to interactively explore how multiple model outputs depend on two selected model inputs. The contour profiler allows us to set limits on the model outputs to help define infeasible and feasible regions; the shape of these shaded regions is dependent on the functional form of the multi-dimensional metamodel. We can also visualize the model output landscape using prediction profilers that show vertical cross sections (SAS Institute 2015). These vertical projections show each model input's effect on the model output; these effects are colored such that green improves the output while red degrades it. These colors allow us to clearly identify the key tradable variables; when there is a color contrast between green and red within a model input's vertical cross section, we must make a trade by accepting an improvement in one model output while accepting a degradation in another.

In addition to identifying the key tradable variables, our dashboard uses an optimization algorithm to find a solution that balances the established model output limits with a weighted desirability function; the desirability function normalizes the model output scale. When our solution does not meet one or more of the model output limits, we then can look to the vertical projections to identify the model inputs that have the highest impact on the unfeasible model outputs; changing these model inputs may result in a feasible solution. If we cannot change a model input to find a feasible solution we then must trade off infeasible model outputs limits in order to arrive at a viable system variant. We can arrive at a variety of viable system variants by setting different model output limits and weights to each of them.

To demonstrate our methodology, we consider a representative use case that involves an opportunity to invest in new technologies that will increase the capabilities of a U.S. Army Infantry Squad. The squad "system" is the collection of integrated technologies that enhance the squad's effectiveness (sensors, weapons, exoskeletons, radios, UAVs, robots, and body armor). This use case provides ample opportunity to highlight tradeoffs across multiple types of costs, performance, schedule, and risk considerations. It also allows for a wide variety of solutions/alternatives that are comprised of different combinations of system components.

2.2 Simulation Base Case / Inputs and Outputs

In order to evaluate the Army Infantry Squad system in an operational context, we use an agent-based simulation called Map Aware Non-uniform Automata (MANA). MANA is a stochastic, agent-based, time-stepped simulation developed by the New Zealand Defense Technology Agency (Lauren and Stephen 2002; McIntosh 2009). MANA is a low-resolution simulation of combat, intended to "capture only enough physics as is necessary." For example, range-probability pairs are used to capture sensor, weapon and communication device effectiveness, rather than attempting to explicitly simulate the physics involved. MANA has been used for numerous studies of military operations (Sanchez and Lucas 2002).

Once the MANA base case was developed, we designed and executed an experiment that consisted of 38 inputs (factors) and 40 outputs (responses); see MacCalman et al. (2015a,b) for more details. For this paper, however, we will illustrate our techniques with just a small subset of the original inputs and outputs consisting of four inputs and six output measures. Each input (potential improvement) was mapped to real world system and sub-system components, and represents a decision factor in our experiment. The four inputs we refer to in this paper are the *Sensor Detection Range*, *Sensor Classification Range*, *Rifle Range* for the M4 rifle, and *Radio Delay*. The first three inputs are varied in the experiment with multipliers that represented the increase in capability, from the baseline. For example, in Figure 3, the lower and upper bounds for *Sensor Classification Ran*ge and *Rifle Range* are 1.0 and 2.0, respectively, indicating that these ranges vary from 100% to 200% of the baseline capability.

Our use case employs the following output measures to evaluate each alternative: *Awareness* (a measure of how well Blue (friendly forces) is aware of the location of Red (enemy forces) over the interval [0,1]), *Lethality* (a measure of how quickly Blue is able to inflict casualties upon Red over the interval [0,1]), *Casualties* (Blue), and the physical weights of the sensors, rifle, and radio, respectively. High values of *Awareness* and *Lethality* are desirable, as are low values of the other output measures.

3 DASHBOARD COMPONENTS

The basis of our dashboard is the set of six metamodels that relate four inputs to the six responses. Simple polynomial metamodels do a reasonable job of characterizing the model behavior in this example, although other metamodel forms can be readily incorporated if they are more suitable. We implement our dashboard in JMP (SAS Institute 2015) because of its built-in, interactive capabilities, but similar dashboards could be constructed using other programming languages.

3.1 Prediction Profiler Dashboard Component

The prediction profiler component (Figure 1) contains a color-coded grid. The row and column with white backgrounds relate to desirability functions, while each of the other columns represents a model input and each of the other rows represents a response. Every other cell in the grid shows a cross section of the row's response as the column factor changes, holding all other factors at designated levels. Two key features of the prediction profiler are: (i) it identifies tradable variables with a color profiler algorithm, and (ii) it optimizes solutions to find input settings that perform well across multiple responses.

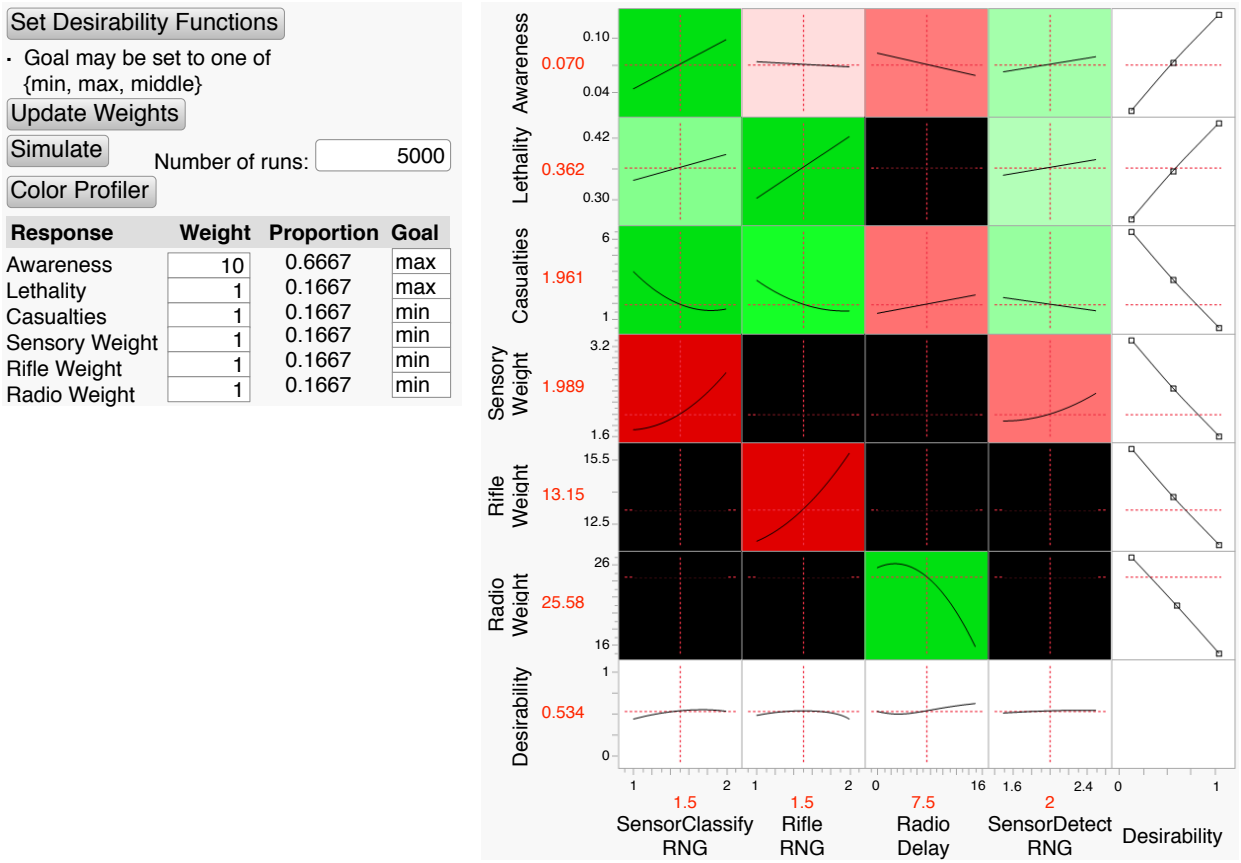


Figure 1: Prediction profiler dashboard component showing vertical cross sections.

3.1.1 Color Profiler Algorithm

The lowest section on the dashboard menu (Figure 1, left) allows the user to specify whether to maximize, minimize, or seek a specified target for each of the responses, and to weight the responses according to their relative importance. Except for the last row and column, the grid cells in the prediction profiler are color coded so that green indicates a positive impact of the input on the response, red indicates a negative impact, and black indicates no impact. A color gradient is automatically applied so the cells with higher impacts are darker and those with lower impacts are lighter, based on the magnitude of the response change between the low and high settings of the model input.

Within each column, the responses that have opposite green and red colors show where the key tradable variables are with respect to each model input. For example, in Figure 1, the column that represents *RifleRNG* has a red cell for *Rifle Weight* and green cells for the *Casualties* and *Lethality* responses. Increasing the rifle range has a positive impact on our operational measures (*Casualties* and *Lethality*) but has a negative impact on a physical consideration (*Rifle Weight*). Therefore, we must trade off weight to achieve a higher operational effectiveness. The model input columns are ordered from left to right in decreasing order of impact across all responses (holding other factors at their mean levels). In this way, we can more readily identify the most impactful model inputs across all responses.

3.1.2 Multiple Response Optimization

The prediction profiler component has a built-in optimization feature that allows the user to specify a weighted desirability function for each response and find a balanced solution (SAS Institute 2015; Myers et al. 2009). The desirability function translates the response scale to a value between nearly zero and nearly one, where zero indicates the least desirable value and one indicates the most desirable value. The user specifies a response goal to maximize, minimize, or achieve a specified target value for each response. Figure 2 shows three types of desirability functions that translate response values along the vertical axis to desirability values along the horizontal axis.

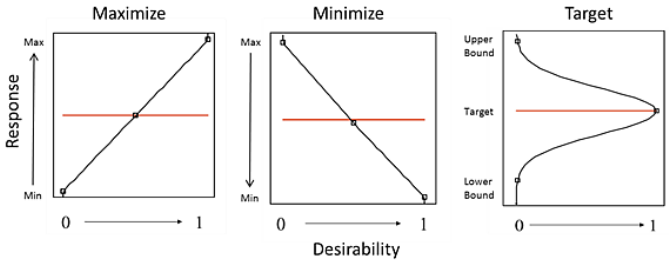


Figure 2: Desirability function types (adapted from SAS Institute, 2015). The red line indicates the response outcome for the current model input settings. The three points on each desirability function can be manually moved to other values within the region, changing the shapes of the desirability functions.

The purpose of desirability transformation is to allow the user to specify the returns to scale along the range of response outcomes, and to establish a common scale across all responses. The common scale allows us to aggregate all responses using a weighted total desirability function. The total desirability function, D^*, has the following form:

$$D^* = \frac{1}{k}[w_1 ln(d_1) + w_2 ln(d_2) + \cdots + w_k ln(d_k)], \quad (1)$$

where k is the number of responses, d_k is the desirability function for response k, and w_k is the importance weight for response k. Equation (1) is the objective function for the optimization algorithm that finds desirable solutions across multiple responses.

3.2 Contour Profiler Dashboard Component

The contour profiler dashboard component shows a collection of contour profilers categorized within different output metric domains. Each contour profiler has one or more responses that pertain to each domain category. Each profilers shows a two-dimensional projection of responses. The two dimensions are displayed along the vertical and horizontal axis and represent two model inputs. The crosshairs inside each contour profiler reflect the current settings for the model inputs. A slider bar above each contour profiler allows the user to set low and high limits (constraints) on the response. These limits represent desired effectiveness and constraints that shade the profilers to indicate infeasible areas of the design space. If the crosshairs fall within a response's shaded region, then the current model input settings will not satisfy the desired effectiveness or constraint for that response. The left portion of Figure 3 shows a screenshot of the contour profiler dashboard component, with the following output domains: Situational Awareness, Lethality, Casualties, and Weight. The first three domains have a single response, while the fourth (Weight) has three responses for each of three components. The top right of Figure 3 shows an expanded version of the floating control panel that allows the user to specify the two model inputs that are shown in each of the contour profilers, and set the levels for all inputs using slider bars. By selecting different model inputs, we can see different parts of the multi-dimensional response surfaces. The bottom right shows a consolidated listing of the responses and their position within ranges either observed from the experiment, or restricted by the user; these listings also appear above their associated contour profiler graph components, but are shown separately for greater readability.

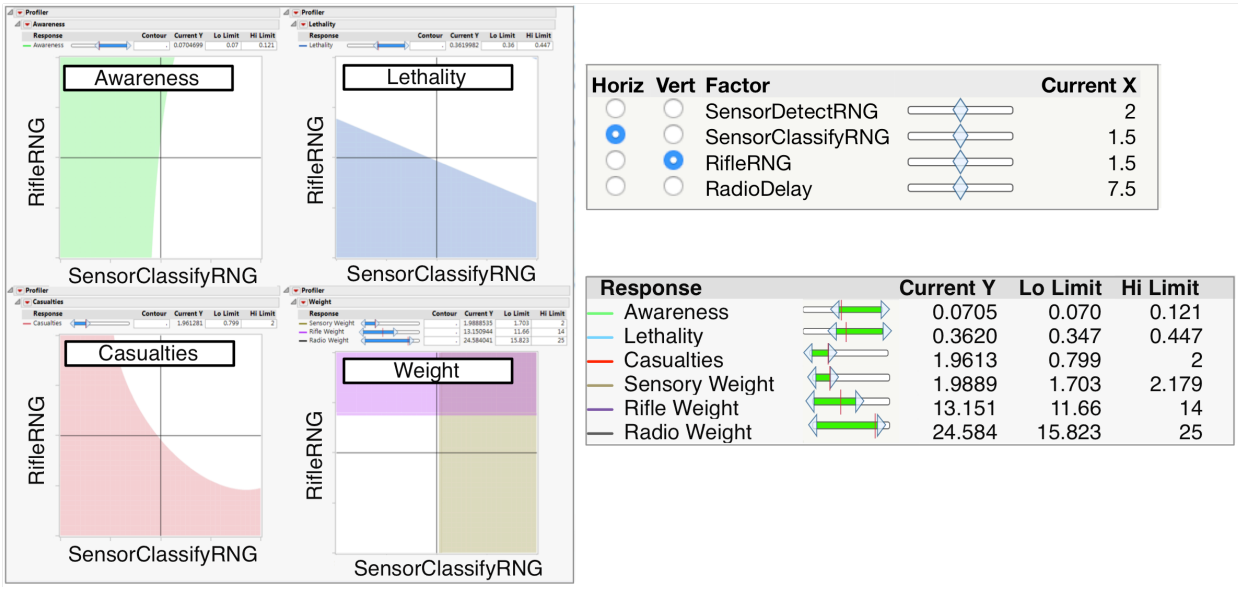


Figure 3: Contour profiler dashboard components.

3.3 Monte Carlo Filtering Component

The purpose of the Monte Carlo Filtering component is to find a collection of feasible alternatives that satisfy all response limits specified in the contour profiler component. A powerful benefit of the response metamodels is that they can act as surrogates to the simulation. Rather than having to run a lengthy simulation to obtain the results of a new system configuration, we can leverage the metamodels to obtain

approximate results in a matter of seconds instead of hours, days or weeks. These approximations can save a tremendous amount of time, especially for a system design problem that relies on several complex models with lengthy run times. The Monte Carlo Filtering dashboard component allows the user to generate thousands of system design configurations using a Monte Carlo simulation. Each simulation run creates a unique system design alternative by drawing a uniform random variate between the low and high settings of each model input. Within the Prediction profiler dashboard component, the user enters the number of simulations and presses the "Simulate" button (see Figure 1, left). A scatter plot matrix appears with a floating response data filter. Each dot in the scatter plot matrix represents a single system alternative. The user can apply the data filters to the responses in order to eliminate alternatives (dots) from the scatter plot. In addition, the user can import and export the response limits set in the contour profiler. After exporting the response limits from the contour profiler to the scatter plot, the alternatives that remain become the feasible set that resides within the white region of the contour profilers. The more Monte Carlo simulations the user specifies, the more solutions there will be within the white region. The end result is a reduced set of viable system variants that satisfy all desired effectiveness constraints. Figure 4 shows a screen shot of the scatterplot matrix for the Monte Carlo Filtering component (on the left) along with the adjustable filtering ranges (on the right). In an interactive mode, the scatterplot updates as the filters are adjusted, showing only those points that simultaneously satisfy all filters ranges.

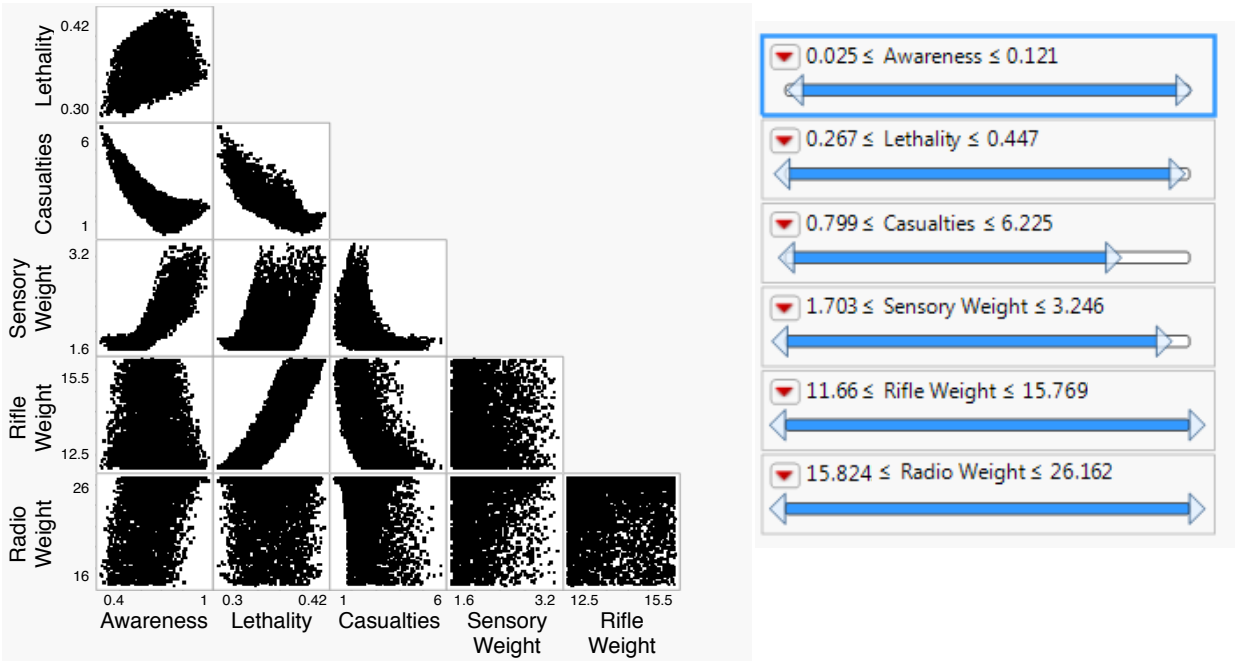


Figure 4: Monte Carlo filtering component, for 5000 random alternatives evaluated via metamodels.

4 VIABLE VARIANT EXPLORATION

We will now walk through potential steps to find viable system variants. Our goal is a set of alternatives, rather than a single solution.

4.1 Step 1: Set Acceptable Response Ranges

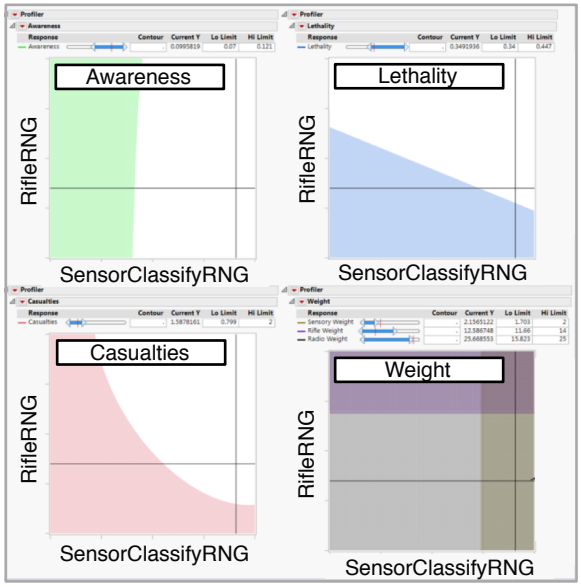
Initially, our contour profilers have no set ranges for the responses. When using the dashboard to find viable variants, we first set the desired constraints on the responses. These are not hard constraints, but are used to shade the infeasible regions on the contour profiler components.

4.2 Step 2: Set Response Importance Weights and Run the Optimization Algorithm

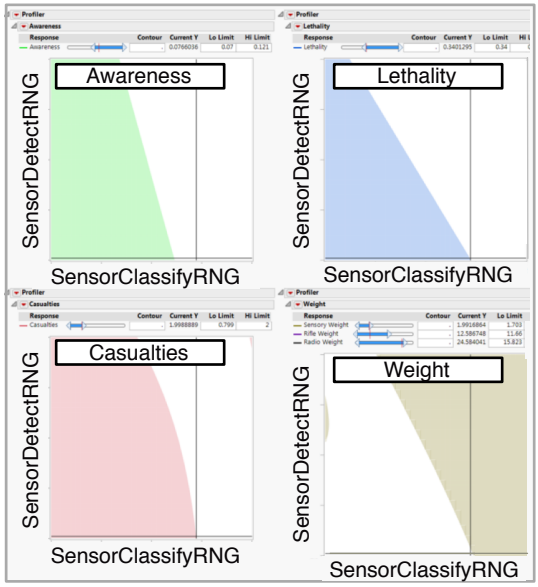
At the left of the Prediction profiler component in Figure 1, the user can specify the importance weights for each response. A system design problem has several stakeholder views and scenarios that prioritize the responses differently. The dashboard allows the user to explore system design solutions with different response prioritization schemes to account for different stakeholder views and scenarios. To run the optimization algorithm, the user clicks the red triangle at the top next to the Prediction Profiler title and selects "Maximize Desirability." After the optimization algorithm completes, the dashboard displays the input settings of the solution. Note that "optimization" is a bit of a misnomer. Even with desirabilities that are all monotonic, slight changes in the desirability functions can change the input settings associated with the maximum desirability solution. The purpose of this step is to generate an alternative that can be added to a set of candidate system variants (if feasible), or explored further or discarded (if infeasible).

4.3 Step 3 (Optional): Explore Changes to Model Input Settings to Find Feasible Solutions

Because the contour profilers are linked to the prediction profilers, we can use them to identify where the optimized solution does not meet the desired effectiveness or feasibility constraints; this occurs where some contour profiler crosshairs fall in a shaded region. Figure 5(a) shows the contour profiler dashboard component with the optimized solution from Step 2. The crosshairs are positioned over two shaded regions, indicating that this solution does not satisfy the *Radio Weight* and *Sensor Weight* responses.



(a) Feasibility of alternatives, shown for RifleRNG vs. SensorClassifyRNG, based on "optimal desirability" solution.



(b) Feasibility of alternatives, shown for SensorDetectRNG vs. SensorClassifyRNG, after two manual factor setting changes.

Figure 5: Two examples of contour profiler dashboard components. In (a), the alternative is feasible in three of the four domains, but violates two constraints in the fourth (lower right). In (b), the contour profiles have changed after manually manipulating the *RadioDelay* and *SensorDetectRNG* factor settings; the crosshairs are now barely feasible (according to the metamodel) in all four domains.

The prediction profiler identifies the model inputs that have the highest impact on the responses, so we can use it to decide which model input to change. Model inputs represent value properties of a system block. As a result, we must consider the practical design implications of changing the value properties

that coincide with a model input change. We may find that the highest impacting model input represents a value property that is too costly or impractical to change.

The prediction profiler in Figure 1 indicates that *RadioDelay* is the model input that has the highest impact on *Radio Weight.* A floating window (not shown) allows us to manually change the two dimensions shown in the contour profilers, using slider bars. Changing the vertical axis to *RadioDelay* and moving the slider bar in the floating window allows us to find a solution that remains feasible for the *Awareness*, *Lethality*, and *Casualties* domains, and meets the *Radio Weight* limit, although it is still violates the *Sensor Weight* constraint. The prediction profiler is automatically updated based on the current input settings, and indicates that the inputs with the greatest effects on *Sensor Weight* are *SensorDetectRNG* and *SensorClassifyRNG*. Reducing the *SensorClassifyRNG* to 1.7 in the contour profiler satisfies the *Sensor Weight* limit, without causing violations to other response constraints. Figure 5(b) shows the contour profiler after these changes to *RadioDelay* and *SensorClassifyRNG* have been made. According to the metamodels, the alternative corresponding to the crosshairs is now (barely) feasible in all dimensions.

The interactive approach in this step is not always a good idea. For example, we must have a very small number of factors—if not, then a trial-and-error approach for searching through the landscape will be time-consuming and ineffective. It is precisely for this reason that we advocate designed experiments, instead of trial-and-error approaches, in the first place! We must also have fairly simple metamodels, unless responses are all highly correlated so that systems that perform well on one metric perform well on all. If we have strong interactions, then the question of whether changing certain inputs improves or degrades certain performance measures depends on the settings of other system components.

4.4 Step 4 (Optional): Trade Off Infeasible Response Limits

Figure 5(b) shows a feasible solution, where all contour profiler crosshairs are in the white region, and therefore we may not need to trade off responses. (A cautionary note: the constraint boundaries do not account for the uncertainty in metamodel estimates.) When no viable variant solution is found by changing model inputs, decide which shaded region response constraint must be relaxed to achieve a feasibility. Modifying the acceptable response ranges can also be useful if new information comes to light. For example, if a new composite material means that other gear carried by the Soldiers becomes lighter, that could mean that the *Sensory Weight* restriction could be increased to (say) 2.25 pounds from 2.0 pounds. The desired end result is a white region within each contour profiler domain that represents a set of viable system variants that (according to metamodels) satisfy all effectiveness constraints.

4.5 Step 5: Generate Viable Variant Solution Candidates using Monte Carlo Filtering

In order to acquire a set of viable variants within the white region of the contour profilers, we use the Monte Carlo Filtering component. Figure 4 is a scatterplot matrix that shows the alternatives generated by the Monte Carlo simulations. A similar matrix (not shown) is generated after the contour profiler response limits are exported to the data filter. Out of the 5,000 simulations summarized by Figure 4, only four alternatives remain when the filters are applied. After running 100,000 simulations with the same response limits and filters, 69 alternatives remain.

4.6 Step 6: Confirm Viability of Solution Candidates

At this stage, it is important to assess the quality of the remaining metamodel-based alternatives by making confirmation runs. If alternatives are close to the edge of the feasible region in the metamodeling domain, then even small amounts of lack-of-fit from the metamodels may mean they are truly infeasible. If alternatives that meet the response limit filters are restricted to small regions of the input space, these may be regions where the overall metamodels do not fit as well. Similarly, the simplification done to

keep the number of metamodel factors small enough to allow for some interactive manipulation means that omitted factors might play an important role.

Confirmation runs may be made in different ways. We use the following for illustration purposes: (i) explore the 69 alternatives for the four metamodel factors while setting the other 34 factors at their baseline (middle) levels; (ii) explore the 69 alternatives while treating the other 34 original factors as a single factor at two levels, all simultaneously set at "maximum capability" or "minimum capability" according to the analyst's opinion; (iii) explore using a resolution III fractional factorial (64 design points) over the other 34 original factors for some or all of the 69 alternatives. Option (ii) provides an optimistic and pessimistic case for each of the 60 alternatives, and indicates whether or not it suffices to focus on the four most important performance drivers, or whether the metamodels should be expanded. Option (iii) can require more design points (and run time) than our original experiment, but has the potential to yield greater variety in the set of viable alternatives.

Our three confirmation experiments (each based on 100 replications) result in 32, 60, and 670 variants, respectively, where the average performance measures meet all six response constraints. The *Casualties* metamodel performs poorly for (ii) and (iii); it tends to underestimate the average casualties from the simulation, and causes more alternatives to be eliminated than the other five responses. Once the confirmation runs are tested, the revised set of alternatives is a set of viable system variants that satisfy the specified constraints of the responses. In order to narrow down the reduced set of solutions further, we can select the most affordable solution, the solution with the least amount of variation, or use other qualitative criteria such as the ease of implementing the solution.

One way to acquire additional viable variants is to repeat steps 1-6 with different sets of response limits and importance weights, representing the perspectives of different stakeholders.

5 CONCLUSIONS AND FUTURE WORK

This paper outlines a process to generate viable variant solutions for tradeoff analysis. It focuses on how to illuminate tradeoffs among multiple simulation outputs by leveraging designed experiments and a dynamic dashboard. This work is essential for engineers and analysts working in the DoD acquisition process, who must use models and simulations from multiple communities of interest to generate alternatives. The need to simultaneously satisfy requirements from multiple domains (such as operational effectiveness, physical feasibility, manufacturability, reliability, and life-cycle cost), by choosing appropriate settings for a large number of system component characteristics, is a very complex problem. We use DOE to specify combinations of simulation inputs that facilitate metamodel creation after the simulation runs are complete. The resulting metamodels identify the system design drivers—those factors having the highest impact on the model output performance—and help focus tradespace exploration in areas that show the promise of generating viable solutions. A dynamic dashboard illustrates an interactive approach to variant exploration that can help identify viable variants and illuminate trade decisions.

This process further allows us to perform additional designed experiments within the region we find in the dashboard in order to confirm the results and develop more detailed metamodels, much like we do during response surface design methods. This adds fidelity to our analysis and allows for more refined analysis of variants in the region or regions of interest. Although this process helps to generate viable variant solutions for tradeoff analysis for numerous tradeoff efforts, its application is constrained by the availability of appropriate models and simulations—as well as the time required to create, verify, and validate a base case and conduct the simulation runs required to perform a designed experiment.

Efforts are under way to include this methodology into a tradespace tool set for the DoD acquisition community. Future work in this area includes assessing the scalability of the process to systems with hundreds to thousands of factors/attributes. Designs are already available (see Kleijnen 2015 or Sanchez and Wan 2015 for references)—but unless the responses are relatively simple, an interactive search approach is problematic if the number of key drivers becomes much larger than our illustration. One-at-a-time changes to inputs in the contour profiler will be ineffective if there are many key drivers and strong

interactions, just as one-at-a-time variation from a baseline is far less effective than a designed experiment. Consequently, scaling the process means that we will rely less on using the interactive aspects of a dashboard to identify good solutions, than to visualize tradeoffs associated with minor changes to specific solutions. Other methods like subset selection might be used to augment (or replace) the "maximum desirability" approach for generating potential starting solutions. Nonetheless, this research provides a tangible benefit to those seeking to design future resilient systems.

ACKNOWLEDGMENTS

U.S. Department of Defense Distribution Statement: Approved for public release; distribution is unlimited. The views expressed in this document are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This work was supported in part by the U.S. Army Engineering Research and Development Center. This paper updates and extends the tradespace visualization section of MacCalman et al. (2015b). Thanks to the Cadets at the United States Military Academy who participated in the study: Coleman Grider, Robert Hill, Thomas Jaeger, and Hunter Wood.

REFERENCES

Barton, R. R. 2015. "Tutorial: Simulation Metamodeling." In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 1765-1779. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Grayson, J., and S. Gardner. 2015. *Building Better Models with JMP Pro*. Cary, North Carolina: SAS Institute Inc.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer-Verlag.

Kleijnen, J. P. C., S. M. Sanchez, T. W. Lucas, and T. M. Cioppa. 2005. "A User's Guide to the Brave New World of Designing Simulation Experiments." *INFORMS Journal on Computing* 17(3): 263–289.

Kleijnen, J. P. C. 2015. *Design and Analysis of Simulation Experiments*. 2nd ed. New York: Springer-Verlag.

Kuhn, M., and K. Johnson. 2013. *Applied Predictive Modeling*. New York: Springer-Verlag.

Lauren, M., and R. Stephen. 2002. "Map-Aware Non-uniform Automata (MANA)—A New Zealand Approach to Scenario Modelling." *Journal of Battlefield Technology* 5(1): 27–31.

MacCalman, A., H. Vieira, and T. Lucas. 2016. "Second-Order Nearly Orthogonal Latin Hypercubes for Exploring Stochastic Simulations." *Journal of Simulation* advance online publication, doi:10.1057/jos.2016.8.

MacCalman, A. D., H. Kwak, M. McDonald, and S. Upton. 2015a. "Capturing Experimental Design Insights in Support of the Model-Based System Engineering Approach." *Procedia Computer Science* 44: 315-324.

MacCalman, A., H. Kwak, M. McDonald, S. Upton, C. Grider, R. Hill, H. Wood, and P. Evangelista. 2015b. "Illuminating Tradespace Decisions Using Efficient Experimental Space-Filling Designs for the Engineered Resilient System Architecture." Technical Report No. DSE-R-1501, United States Military Academy, West Point, New York.

McIntosh, G. C. 2009. *MANA-V (Map Aware Non-uniform Automata-Vector) Supplementary Manual.* New Zealand: Defence Technology Agency.

Myers, R. H., D. C. Montgomery, and C. M. Anderson-Cook. 2009. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. 3rd ed. New Jersey: Wiley.

Sanchez, S. M., and T. W. Lucas. 2002. "Exploring the World of Agent-Based Simulation: Simple Models, Complex Analyses." In *Proceedings of the 2002 Winter Simulation Conference*, edited by E.

Yücesan, C. Chen, J. L. Snowdon, and J. Charnes, 116–126. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sanchez, S. M. 2015. "Simulation Experiments: Better Data, Not Just Big Data." In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 800-811. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Sanchez, S. M., P. J. Sanchez, and H. Wan. 2014. "Simulation Experiments: Better Insights by Design." In *Proceedings of the 2014 Summer Simulation Conference*. San Diego, California: The Society for Modeling & Simulation International.

Sanchez, S. M., and H. Wan. 2015. "Work Smarter, Not Harder: A Tutorial on Designing and Conducting Simulation Experiments." In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 1795-1809. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

SAS Institute. 2015. *JMP 12 Profilers*. Cary, North Carolina: SAS Institute Inc.

AUTHOR BIOGRAPHIES

ALEX D. MACCALMAN is an Army Special Forces Officer in the Operations Research System Analyst Functional Area. He has a B.S in Engineering Management from the United States Military Academy, an M.S. in Operations Research, and a Ph.D in Simulation Analysis from the Naval Postgraduate School. He is currently the Systems Engineering Program Director at the United States Military Academy. His email is alex.maccalman@gmail.com.

SUSAN M. SANCHEZ is a Professor in Operations Research at the Naval Postgraduate School, and Co-Director of the Simulation Experiments & Efficient Designs (SEED) Center for Data Farming. She also holds a joint appointment in the Graduate School of Business & Public Policy. She has a B.S. in Industrial & Operations Engineering from the University of Michigan, and a Ph.D. in Operations Research from Cornell. She has long been active in within the simulation community, including the WSC Board of Directors. Her web page is http://faculty.nps.edu/smsanche/, and her e-mail address is ssanchez@nps.edu.

MARY L. MCDONALD is a Research Associate in the SEED Center for Data Farming at the Naval Postgraduate School. She has a B.A. in Mathematics from Northwestern University, and an M.S. in Applied Mathematics from the Naval Postgraduate School. She teaches probability and statistics and has almost 20 years of experience in military simulation and analysis. Her email address is mlmcdona@nps.edu.

SIMON R. GOERGER is the Director for the Institute for Systems Engineering Research (ISER), U.S. Army Engineer Research and Development Center (ERDC). He received his B.S. from the United States Military Academy (USMA), his M.S. National Security Strategy from the National War College, and his M.S. in Computer Science and his Ph.D. in Modeling and Simulation both from the Naval Postgraduate School. He is a Retired Colonel from the U.S. Army, where his appointments included Director of the Operations Research Center of Excellence in the Department of Systems Engineering at USMA. His email address is Simon.R.Goerger@erdc.dren.mil.

ANDREW T. KARL is a statistician working with Adsurgo LLC. He received his B.A. in mathematics from the University of Notre Dame and his Ph.D. in statistics from Arizona State University. His research and teaching areas include mixed models, experimental design, text mining, data mining, and data visualization. His email address is andrew.karl@adsurgo.com.