Cite as:
Joseph Krall, Tim Menzies, and Misty Davies.
"Learning the Task Management Space of an Aircraft Approach Model."
Proceedings of the 2014 AAAI Symposium on Formal Verification and Modeling in Human-machine Systems.
Palo Alto, CA. March, 2013.

Learning the Task Management Space of an Aircraft Approach Model

Joseph Krall and **Tim Menzies**
Lane Department of CS&EE
West Virginia University, WV, USA
(kralljoe,tim.menzies)@gmail.com

Misty Davies
Intelligent Systems Division
NASA Ames Research Center, CA, USA
misty.d.davies@nasa.gov

Abstract

Validating models of airspace operations is a particular challenge. These models are often aimed at finding and exploring safety violations, and aim to be accurate representations of real-world behavior. However, the rules governing the behavior are quite complex: nonlinear physics, operational modes, human behavior, and stochastic environmental concerns all determine the responses of the system. In this paper, we present a study on aircraft runway approaches as modeled in Georgia Tech's Work Models that Compute (WMC) simulation. We use a new learner, Genetic-Active Learning for Search-Based Software Engineering (GALE) to discover the Pareto frontiers defined by cognitive structures. These cognitive structures organize the prioritization and assignment of tasks of each pilot during approaches. We discuss the benefits of our approach, and also discuss future work necessary to enable uncertainty quantification.

The Motivation—Complexity in Aerospace

Complexity that works is built of modules that work perfectly, layered one over the other. –Kevin Kelly

The National Airspace System (NAS) is complex. Each airplane is an intricate piece of machinery with both mechanical and electrical linkages between its many components. Engineers and operators must constantly decide which components and interactions within the airplane can be neglected. As one example, the algorithms that control the heading of aircraft are usually based on linearized versions of the actual (very nonlinear) dynamics of the aircraft in its environment. (Blakelock 1991) Each airplane must also interact with other airplanes and the environment. For instance, weather can cause simple disruptions to the flow of the airspace, or be a contributing factor to major disasters. (NTSB 2010) Major research efforts are currently focused on models and software to mitigate weather-based risks. (Le Ny and Balakrishnan 2010)

The glue for these interacting airspace systems consists primarily of people. Pilots and air traffic controllers are the final arbiters and the primary adaptive elements; they are expected to compensate for weather, for mechanical failures, and for others' operational mistakes. They are also the scapegoats. Illustratively, examine the failure of a software system at the Los Angeles Air Route Traffic Control Center on September 14, 2004. (Geppert 2004) Voice communications ceased between the controllers and the 400 aircraft flying above 13,000 feet over Southern California and adjacent states. During the software malfunction, there were five near-misses between aircraft, with collisions prevented only by an on-board collision detection and resolution system (TCAS). At the time, the FAA was in the process of patching the systems. As often happens in software-intensive systems, the intermediate 'fix' was to work around the problem in operations—the software system was supposed to be rebooted every 30 days in order to prevent the occurrence of the bug. The human operators hadn't restarted the system, and they were blamed for the incident.

If the current state of airspace complexity causes palpitations, experts considering what might happen in the planned next generation (NextGen) airspace can be excused for full-fledged anxiety attacks. The future of the NAS is more heterogeneity and more distribution of responsibility. We are seeing a switch to a *best-equipped best-served* model—airlines who can afford to buy and operate equipment can get different treatment in the airspace. One example is the advent of Required Navigation Performance (RNP) routes, in which aircraft fly tightly-controlled four-dimensional trajectories by utilizing GPS data. With GPS, an aircraft can be cleared to land and decscend from altitude to the runway in a *Continuous Descent Arrival (CDA)*; these approaches save fuel and allow for better-predicted arrival times. However, at airports with these approved routes, controllers must work with mixed traffic—airplanes flying CDA routes and airplanes flying traditional approaches. In the future, the airspace will also include Unmanned Aerial Systems (fully-autonomous systems, and also systems in which a pilot flies multiple aircraft from the ground), and a wider performance band for civil aircraft.

The overall traffic increase is leading to software-based decision support for pilots and controllers. There is an active (and sometimes heated) discussion about just how much authority and autonomy should remain with people versus being implemented in software. Decisions about where loci of control should reside in the airspace is an example of a

wicked design problem (Rittel 1984; Hooey and Foyle 2007) as evidenced by the following criterion:

- Stakeholders disagree on the problem to solve.
- There are no clear termination rules.
- There are 'better' or 'worse' solutions, but not 'right' and 'wrong' solutions.
- There is no objective measure of success.
- The comparison of design solutions requires iteration.
- Alternative solutions must be discovered.
- The level of abstraction that is appropriate for defining the problem requires complex judgments.
- It has strong moral, political, or professional dimensions that cannot be easily formalized.

In this paper we will first discuss a simulation that is designed to study human-automation interaction for CDAs. We will then overview the current state-of-the-art for uncertainty quantification within this type of complex system, and focus on techniques for exploring the Pareto Frontier. In the next section, we will explain and demonstrate a new technique (GALE) for quickly finding the Pareto Frontier for 'wicked' problems like those we study in our use case. We will conclude by overviewing our future plans.

How Do Pilots and Air Traffic Controllers Land Planes? Our Test Case and Its Inputs

In this paper, we use the CDA scenario within the Georgia Institute of Technology's *Work Models that Compute* (WMC). WMC is being used to study concepts of operation within the (NAS), including the work that must be performed, the cognitive models of the agents (both humans and computers) that will perform the work, and the underlying nonlinear dynamics of flight. (Kim 2011; Pritchett, Christmann, and Bigelow 2011; Feigh, Dorneich, and Hayes 2012) WMC and the NAS are *hybrid systems*, governed both by continuous dynamics (the underlying physics that allows flight) and also discrete events (the controllers' choices and aircraft modes). (Pritchett, Lee, and Goldsman 2001) Hybrid systems are notoriously difficult to analyze, for reasons we will overview in the next section.

WMC's cognitive models are multi-level and hierarchical, (Kim 2011) with:

- Mission Goals at the highest level, such as *Fly and Land Safely*, that are broken into
- Priority and Values functions such as *Managing Interaction with the Air Traffic System*. These functions can be decomposed into
- Generalized Functions such as *Managing the Trajectory*, that can still be broken down further into
- Temporal Functions such as *Controlling Waypoints*.

In this paper, we present some preliminary results in which we have varied four input parameters to WMC in order to explore their effects on the simulation's behavior. We list these parameters in bold, and then describe them. The **Scenario** is a variable within the CDA simulation with four values. In the *nominal* scenario, the aircraft follows the ideal case of arrival and approach, exactly according to printed charts, with no wind. In the *late descent* scenario, the air traffic controller delays the initial descent, forcing the pilots to quickly descend in order to 'catch up' to the ideal descent profile. In the third, *unpredicted rerouting*, scenario, the air traffic controller directs the pilot to a waypoint that is not on the arrival charts, and from there returns the pilot to the expected route. In the final scenario, the simulation creates a *tailwind* that the pilot and the flight deck automation must compensate for in order to maintain the correct trajectory. The *late descent*, *unpredicted rerouting*, and *tailwind* scenarios all have further variants, modifying the times at which the descent is cleared, the waypoint that the plane is routed to, and the strength of the tailwind, respectively.

Function Allocation is a variable that describes different strategies for configuring the autoflight control mode, and has four different possible settings. A pilot may have access to guidance devices for the lateral parts (*LNAV*) and the vertical parts (*VNAV*) of the plane's approach path. Civil transport pilots are likely to have access to a Flight Management System (FMS), a computer that automates many aviation tasks. In the first Function Allocation setting, which is *highly automated*, the pilot uses LNAV/VNAV, and the flight deck automation is responsible for processing the air traffic instructions. In the second, *mostly automated*, setting, the pilot uses LNAV/VNAV, but the pilot is responsible for processing the air traffic instructions and for programming the autoflight system. In the third setting, the pilot receives and processes the air traffic instructions. The pilot updates the vertical autoflight targets; the FMS commands the lateral autoflight targets. This setting is the *mixed-automated* function allocation setting. In the final, *mostly manual*, setting, the pilot receives and processes air traffic instructions, and programs all of the autoflight targets.

The third parameter we are varying in this paper is the pilots' **Cognitive Control Modes**. There are three cognitive control modes implemented within WMC: *opportunistic*, *tactical*, and *strategic*. In the opportunistic cognitive control mode, the pilot does only the most critical temporal functions: the actions "monitor altitude" and "monitor airspeed." The values returned from the altitude and the airspeed will create tasks (like deploying flaps) that the pilot will then perform. In the tactical cognitive control mode, the pilot cycles periodically through most of the available monitoring tasks within WMC, including the confirmation of some tasks assigned to the automation. Finally, in the strategic mode, the pilot monitors all of the tasks available within WMC and also tries to anticipate future states. This "anticipation" is implemented as an increase in the frequency of monitoring, and also a targeted calculation for future times of interest.

Finally, the fourth variable we explore in this paper is **Maximum Human Taskload**: the maximum number of tasks that can be requested of a person at one time. In previous explorations using WMC (Kim 2011), the author chose three different levels: *tight*, in which the maximum number of tasks that can be requested of a person at one time is 3; *moderate*, in which that value is 7; and *unlimited*, in which a person is assumed to be able to handle up to 50 requested tasks at one time. WMC uses a task model in which tasks have priorities and can be active, delayed, or interrupted. (Feigh and Pritchett 2013) If a new task is

passed to a person and that person's maximum taskload has been reached, an active task will be delayed or interrupted, depending on the relative priorities of the tasks that have been assigned. Delayed and interrupted actions may be *forgotten* according to a probability function that grows in the time elapsed since the task was active. For the studies in this paper, we assume that people can handle between 1 and 7 tasks at maximum. (Miller 1956; Cowan 2000; Tarnow 2010)

Our analysis seeks to explore the effects that each of the four variables above has on the following five outputs: the number of forgotten tasks in the simulation (*NumForgotten Tasks*), the number of delayed actions (*NumDelayedActions*), the number of interrupted actions (*NumInterruptedActions*), the total time of all of the delays (*DelayedTime*), and the total time taken to deal with interruptions (*InterruptedTime*). In our results, we refer to these outputs as (*f1 . . . . f5*) and average each of their values across the pilot and the copilot.

In Kim's dissertation (Kim 2011), she primarily studies function allocation and its effect on eight different parameters, including workload and mission performance. In this sense, the WMC model by itself as Kim chose to use it (much less the airspace it is meant to simulate) is 'wicked'. In particular, there is no single measure of success, and there is no agreement as to which of the measures is more important. Kim analyzed all of the combinations of the above four variables, and manually postprocessed the data in order to reach significant conclusions about how the level-of-automation affects each of her eight metrics.

An Overview of Uncertainty Quantification Within Hybrid, Wicked Systems

Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful. –George E.P. Box and Norman R. Draper

Validation is the process by which analysts answer "Did we solve the right problem?" Uncertainty (and risk) quantification is core to the validation of safety-critical systems, and is particularly difficult for wicked design problems. WMC is a tool that is aimed at validating concepts of operation in the airspace. It abstracts some components within the airspace, and approximates other components, and must itself be validated in order to understand its predictive strengths and limitations. Validation efforts can take as a given that WMC's predictions are useful, and be focused on discovering the risks in the concepts of operation (in which case the analysis is usually called risk quantification). Uncertainty quantification within the model is usually focused on comparing the predictions to those we get (or to those we expect to get) in reality. The questions we are asking in each of these two cases are different, but the underlying tools we use in order to analyze them is often the same.

In the case of risk quantification, where we want to validate the concept of operation, we explore the input and output spaces of our models, looking for those that perform better or worse among the many metrics we've chosen to examine. For simulations with long response times, or for which we hope to learn about a broad class of behaviors using relatively few trials, we build a secondary model that is easier to evaluate than the original simulation. Whichever surface we can evaluate, whether it is the original or a secondary model, is called a *response surface*. In the case of uncertainty quantification, where we want to validate our model, we again build a response surface for our model and compare this against the response surface built using real (or expected) behaviors.

A common way of characterizing a response surface is by building a *Pareto Frontier*. A Pareto Frontier occurs when a system has competing goals and resources; it is the boundary where it is impossible to improve on one metric without decreasing another. (Lotov, Bushenkov, and Kamenev 2004) A Pareto Frontier is usually discovered using an optimization methodology. In rare cases, it may be possible to analytically discover the Pareto Frontier—this is unlikely in wicked design problems like those we are studying here. More often, we use a learning technique to discover the Pareto Frontier given concrete trials of the system.

Classical optimization techniques are often founded on the idea that the response surface and its first derivative are Lipschitz continuous everywhere (*smooth*). For smooth surfaces, it is possible to find a response surface that is arbitrarily close to our desired function using polynomial approximations by the Weierstrass Approximation Theorem. (Bartle 1976) For the hybrid, complex, non-linear problems we are studying here, no such guarantee of smoothness exists. Modal variables like the cognitive control modes in WMC usually require combinatorial approaches in order to explore. For other WMC inputs, such as the maximum human taskload, a domain expert might reasonably suspect that there is an underlying smooth behavior. For some WMC inputs we haven't modeled yet, such as flight characteristics of the aircraft or the magnitude of a tailwind, there is almost certainly a smooth relationship, but it may be nonlinear. Classical techniques handle the mix of discrete and continuous inputs by solving a combinatorial number (in the discrete inputs) of optimization problems over the continuous inputs, and then comparing the results across the optimizations in a post-processing step. (Gill, Murray, and Wright 1986) This technique can be computationally very expensive, especially when you consider that continuous optimization techniques are sensitive to local minima (in our nonlinear aerospace problems), and several different input trials should be performed.

Statistical techniques such as Treed Gaussian Processes and Classification Treed Gaussian Processes, can be used to build statistical emulators as the response surfaces for simulators, and have the advantage that they can model discontinuities and locally smooth regions. (Gramacy 2007; He 2012) As a disadvantage, they are limited by computational complexity to relatively few inputs (10s but not 100s). More recent techniques, such as those based on particle filters, can handle significantly many more inputs. (He and Davies 2013)

All of the above techniques have the limitation that they optimize for one single best value. To optimize across several criterion (such as the five we analyze for this paper or

the eight in Kim's thesis) using the above techniques, the analyst usually needs to build a *penalty function*, a formula that is strictly monotonic in improvement across the desired metrics and weights each metric according to its relative value. In this paper, we choose instead to explore the class of multi-objective response surface methods, as detailed in the next section.

GALE: Active Learning for Wicked Problems

Wicked problems have many features; the most important being that no objective measure of success exists. Designing solutions for wicked problems cannot aim to produce some perfectly correct answer since no such definition of correct exists. Hence, this approach to design tries to support effective *debates* by a community over a range of possible answers. For example, different stakeholders might first elaborate their own preferred version of the final product or what is important about the current problem. These preferred versions are then explored and assessed.

The issue here is that there are very many preferred versions. For example, consider the models discussed in this paper. Just using the current models, as implemented by Kim et al. (Kim 2011), the input space can be divided 144 ways, each of which requires a separate simulation. In our exploration, we further subdivide the maximum human taskload to evaluate 252 combinations. Worse yet, a detailed reading of Kim's thesis shows that her 144 input sets actually explore only one variant each for three of her inputs. Other modes would need to be explored to handle:

- Unpredictable rerouting;
- Different tail wind conditions;
- Increasing levels of delay.

If we give three "what-if" values to the above three items then, taken together, these 3*3*3*252 modes*inputs would require nearly 7000 different simulations[1]. This is an issue since, using standard multi-objective optimizers such as NSGA-II (Deb et al. 2000), our models take seven hours to reach stable minima. Hence, using standard technology, these 7,000 runs would take 292 weeks to complete. In principle, such long simulations can be executed on modern CPU clusters. For example, using the NASA Ames multicore supercomputers, the authors once accessed 30 weeks of CPU in a single week. Assuming access to the same hardware, our 7,000 runs might be completed in under ten weeks.

The problem here is that hardware may not be available. The example in the last paragraph (where 30 weeks of CPU time was accessed in one week) was only possible since there was a high priority issue in need of urgent resolution. In the usual case at NASA, researchers can only access a small fraction of that CPU. For example, if there has been some incident on a manned space mission, then NASA enlists all available CPU time for "damage modeling" (which is a large series of "what-if" queries that assess the potential impact of some event). At those times, researchers can access zero CPU for any other purpose.

GALE, short for Geometric Active Learning Evolution, combines spectral learning and response surface methods to reduce the number of evaluations needed to assess a set of candidate solutions. The algorithm is an active learner; i.e. instead of evaluating all instances, it isolates and explores only the most informative ones. Hence, we recommend GALE for simulations of wicked problems. The following notes are a brief overview on GALE. For full details, see (Krall 2014; Krall and Menzies 2014).

Response surface methods (RSM) generate multiple small approximations to different regions of the output space. Multi-objective RSMs explore the Pareto frontier (the space of all solutions dominated by no other). These approximations allow for an extrapolation between known members of the population and can be used to generate approximations to the objective scores of proposed solutions (so after, say, 100 evaluations it becomes possible to quickly approximate the results of, say, 1000 more). Other multi-objective RSMs make parametric assumptions about the nature of that surface (e.g. Zuluaga et al. assume they can be represented as Gaussian process models (Zuluaga et al. 2013)). GALE uses non-parametric multi-objective RSMs so it can handle models with both discrete and continuous variables.

GALE builds its response surface from clusters on the Pareto frontier. These are found via a recursive division of individuals along the principal component found at each level of the recursion[2]. Spectral learners like GALE base their reasoning on these eigenvectors since they simultaneously combine the influences of important dimensions while reducing the influence of irrelevant or redundant or noisy dimensions (Kamvar, Klein, and Manning 2003). Recursion on n individuals stops when a division has less than $\sqrt{n}$ members. At termination, this procedure returns a set of leaf clusters that it calls *best*.

During recursion, GALE evaluates and measures objective scores for a small m number of individuals. These scores are used to check for domination between two partitions of individuals, divided at the some middle point (chosen to minimize the expected variance over each partition) of that level's component. GALE then only recurses on the non-dominated half. That is, the *best* individuals found by GALE are clusters along the Pareto frontier.

GALE is an active learner. During its recursion, when exploring n randomly generated solutions, GALE only evaluates at most $m \log_2(n)$ individuals. One surprising result from our experiments is that GALE only needs to check for domination on only the $m = 2$ most separated individuals along the principal component (which is consistent with Pearson's original claim that these principal components are an informative method of analyzing data (Pearson 1901)).

For reasons of speed, GALE uses a Nyström technique (called FASTMAP) to find the principal component (Faloutsos and Lin 1995; Platt 2005). At each level of its recursion,

[1] To be accurate, there are many more than 7,000 possible simulations, especially if we start exploring fine-grained divisions of continuous space. Regardless of whether or not we need 7,000 or 7,000,000 simulations, the general point of this section still holds; i.e. wicked problems need some way to explore more options faster.

[2] The principle component of a set of vectors shows the general direction of all the vectors (Pearson 1901).

	num eval	f1	f2	f3	f4	f5	s percentiles 50th	(75-25)th
GALE	33	0.8	0.0	0.0	0.0	0.0	82%	0%
NSGAII	4,000	1.2	0.0	0.0	0.0	0.0	84%	0%
SPEA2	3,200	1.0	0.0	0.0	0.0	0.0	84%	1%
Baseline	-	8.2	0.1	0.1	0.2	0.1	100%	0%

Table 1: Raw values for f1, f2, f3, f4, f5 = NumForgottenTasks, NumDelayedActions, NumInterruptedActions, DelayedTime, InterruptedTime, respectively. *Lower* is better.

this technique finds in linear time the *poles* p, q (individuals that are furthest apart) and the approximation to the principal component is the line from p to q. GALE handles continuous and discrete variables by adopting the distance function of Aha et al., which can manage continuous and discrete variables (Aha, Kibler, and Albert 1991).

Ostrouchov and Samatova show that the poles found by FASTMAP are approximations to the vertexes of the convex hull around the individuals (Ostrouchov and Samatova 2005). Therefore, we can use FASTMAP as a response surface method by extrapolating between the poles of the *best* clusters. Given some initial set of individuals, GALE defines the *baseline* to be the median value of all their objectives. (Note that this baseline and initial population are generated only once, and then cached for reuse.) For each cluster $c_i \in$ *best* and for each pole $(p, q) \in c_i$, GALE sorts the poles by their *score* (denoted s) where s is the sum of the distance of each objective from the baseline (and *better* scores are *lower*)[3]. The *best* individuals in leaf clusters are mutated towards their better pole by an amount

$$\forall d \in D, d = d + \Delta \frac{s(p)}{s(q)}$$

where q is the pole with better (and lowest) score, D are the decisions within an individual, and $0 \leq \Delta \leq 1$ is a random variable. GALE grows new solutions using ranges in the mutated population. Numbers are discretized into ten ranges using $(x - min)/((max - min)/10)$. The most frequent range is then found for each feature and new individuals are generated by selecting values at random from within those ranges. GALE then recurses on the new individuals.

GALE's performance has been compared to two standard MOEAs (NSGA-II and SPEA2 (Deb et al. 2000; Zitzler, Laumanns, and Thiele 2001)) on (a) a software process model of agile development (Lemon et al. 2009), as well as (b) a sample of the standard optimization certification problems (Krall 2014; Krall and Menzies 2014). In that study, GALE terminated using 20 to 89 times fewer evaluations. Further, its solutions were usually as good or better than those of NSGA-II or SPEA2. The conclusion from that study was that GALE's RSM was a better guide for mutation than the random search of NSGA-II or SPEA2.

Results

We ran three optimization (GALE, NSGA-II, and SPEA2) algorithms on CDA, and show their results in Table 1. The

[3]To be precise, s is the "loss" measure discussed by (Krall and Menzies 2014), as inspired by (Zitzler and Künzli 2004).

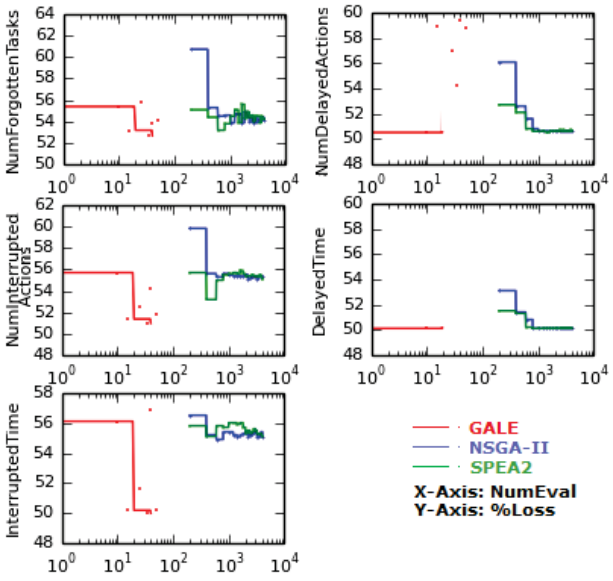

Figure 1: Visualizations of %loss from baseline of objective scores. Number of evaluations are shown on the horizontal axis. Y-axis values show objectives achieved expressed as a ratio of values seen in the baseline; e.g. $y = 50$ means that an optimizer has achieved some objective value that is half that seen in the baseline. Shown as red, blue, green lines is the lowest seen objective score at that particular value along the X-axis; lower values are better.

three algorithm rows of this table indicate the final objective scores of CDA in the f-columns. The Baseline row (the same $\mu = 100$ sized population is used for each algorithm) indicates the starting points for those objectives. These algorithm rows should be compared to the baseline row, and the improvements are easily noticeable.

The previous paragraph addresses the validity of CDA, e.g. if it can be optimized. Now, we turn who can optimize best. When comparing optimizers, we need to compare 1) how well, and 2) how fast. In Figure 1, colored lines represent the best seen improvements of each algorithm (lower is better). In general, each color is evenly matched, except for interrupted time (f5), where the red clearly outperforms the blue and green. Thus, GALE is better on "how well".

As for "how fast", we return to Table 1 to compare the "num eval" of each algorithm. Each evaluation is equivalent to running CDA one time. Note that the average running time of CDA itself is about 8 seconds. This means that GALE (33 evals) can optimize the CDA model in about 4 minutes versus the 7 hours needed by NSGA-II (4000 evals) or SPEA2 (3200 evals). That is, GALE is about 60x faster.

Please note that these results are from running each algorithm only once each. A more complete study is in progress, but due to the recent government shutdown, we were unable to complete our goal of 20 runs of NSGA-II and SPEA2. However, in keeping with the main message of this paper, we note that we could finish all the GALE runs (these are not shown since it makes little sense to compare solo runs

with multiple runs).

Conclusion

In this paper, we've shown that GALE can learn the 'wicked' response surface for an aerospace task management model at similar accuracy and much faster than other similar techniques. Optimization (GALE), explanation (visualizations and charts), and encapsulation (ruleset summarization) are tools that together comprise the validation of models. GALE's fast learning allows us to more thoroughly explore the envelope of behaviors, leading to overall improved validation.

Our immediate next steps involve the thorough data collection of experiments described in this paper, since only the results for N=1 runs of each of NSGA-II, SPEA2 and GALE were detailed. Our further plans are then to improve GALE's reporting suite on the learned results; we'd like to generate succinct rulesets on how best to build test cases with optimal solutions. This will include both ruleset summarization and also validity assertions through re-evaluating the model on individuals generated via the ruleset. Such a ruleset can be used to validate the CDA model itself. In the longer term, we intend to expand the analysis to include more complex and realistic scenarios including a larger number of input parameters evaluated against more output metrics.

Acknowledgements

The work was funded by NSF grant CCF:1017330 and the Qatar/West Virginia University research grant NPRP 09-12-5-2-470. This research was conducted at NASA Ames Research Center. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

References

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Mach. Learn.* 6(1):37–66.

Bartle, R. 1976. *The elements of real analysis*. John Wiley & Sons, second edition.

Blakelock, J. H. 1991. *Automatic Control of Aircraft and Missiles*. Wiley-Interscience.

Cowan, N. 2000. The magical number 4 in short term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences* 24:87–185.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2000. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6:182–197.

Faloutsos, C., and Lin, K.-I. 1995. FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 163–174.

Feigh, K. M., and Pritchett, A. R. 2013. Modeling the work of humans and automation in complex operations. In *51st AIAA Aerospace Sciences Meeting*.

Feigh, K. M.; Dorneich, M. C.; and Hayes, C. C. 2012. Toward a characterization of adaptive systems: A framework for researchers and system designers. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 54(6):1008–1024.

Geppert, L. 2004. Lost radio contact leaves pilots on their own. *IEEE Sepctrum* 41(11):16–17.

Gill, P. E.; Murray, W.; and Wright, M. H. 1986. *Practical Optimization*. Elsevier.

Gramacy, R. B. 2007. tgp: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models. *Journal of Statistical Software* 19(9):1–46. `http://www.jstatsoft.org/v19/i09/paper`.

He, Y., and Davies, M. 2013. Validating an air traffic management concept of operation using statistical modeling. In *AIAA Modeling and Simulation Technologies Conference*.

He, Y. 2012. *Variable-length Functional Output Prediction and Boundary Detection for an Adaptive Flight Control Simulator*. Ph.D. Dissertation.

Hooey, B. L., and Foyle, D. C. 2007. Requirements for a design rationale capture tool to support NASA's complex systems. In *International Workshop on Managing Knowledge for Space Missions*.

Kamvar, S.; Klein, D.; and Manning, C. 2003. Spectral learning. In *IJCAI'03*, 561–566.

Kim, S. Y. 2011. *Model-Based Metrics of Human-Automation Function Allocation in Complex Work Environments*. Ph.D. Dissertation, Georgia Institute of Technology.

Krall, J., and Menzies, T. 2014. GALE: Genetic active learning for search-based software engineering. *IEEE TSE (under review)*.

Krall, J. 2014. *JMOO: Multi-Objective Optimization Tools for Fast Learning*. Ph.D. Dissertation.

Le Ny, J., and Balakrishnan, H. 2010. Feedback control of the National Airspace System to mitigate weather disruptions. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, 2055 –2062.

Lemon, B.; Riesbeck, A.; Menzies, T.; Price, J.; D'Alessandro, J.; Carlsson, R.; Prifiti, T.; Peters, F.; Lu, H.; and Port, D. 2009. Applications of simulation and AI search: Assessing the relative merits of agile vs traditional software development. In *IEEE ASE'09*.

Lotov, A. V.; Bushenkov, V. A.; and Kamenev, G. K. 2004. *Interactive Decision Maps: Approximation and Visualization of Pareto Frontier*. Applied Optimization. Kluwer Academic Publishers.

Miller, G. A. 1956. The magical number seven, plus or minus two: Some thoughts on our capacity for processing information. *Psychological Review* 101(2):343–352.

NTSB. 2010. Weather-related aviation accident study 2003-2007. Technical report, National Transportation Safety Board.

Ostrouchov, G., and Samatova, N. F. 2005. On FastMap and the convex hull of multivariate data: Toward fast and robust dimension reduction. *IEEE Trans. Pattern Anal. Mach. Intell.* 27(8):1340–1343.

Pearson, K. 1901. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine* 2(11):559..572.

Platt, J. C. 2005. FastMap, MetricMap, and Landmark MDS are all Nyström algorithms. In *In Proceedings of 10th International Workshop on Artificial Intelligence and Statistics*, 261–268.

Pritchett, A. R.; Christmann, H. C.; and Bigelow, M. S. 2011. A simulation engine to predict multi-agent work in complex, dynamic, heterogeneous systems. In *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*.

Pritchett, A.; Lee, S.; and Goldsman, D. 2001. Hybrid-system simulation for National Airspace System safety analysis. *Journal of Aircraft* 38(5):835–840.

Rittel, H. 1984. Second generation design methods. In Cross, N., ed., *Development in Design Methodology*. New York: John Wiley and Sons.

Tarnow, E. 2010. There is no capacity limited buffer in the Murdock (1962) free recall data. *Cognitive Neurodynamics* 4:395–397.

Zitzler, E., and Künzli, S. 2004. Indicator-based selection in multiobjective search. In *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, 832–842. Springer.

Zitzler, E.; Laumanns, M.; and Thiele, L. 2001. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou, K. C.; Tsahalis, D. T.; Périaux, J.; Papailiou, K. D.; and Fogarty, T., eds., *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, 95–100. International Center for Numerical Methods in Engineering.

Zuluaga, M.; Krause, A.; Sergent, G.; and Püschel, M. 2013. Active learning for multi-objective optimization. In *International Conference on Machine Learning (ICML)*.