Learning Mitigations for Pilot Issues when Landing Aircraft (via Multi-Objective Optimization and Multi-agent Simulations)

Joseph Krall, Tim Menzies *Member, IEEE*, Misty Davies *Member, IEEE*

***Abstract*—We advocate exploring complex models by combining data miners (to find a small set of most critical examples) and of multi-objective optimizers (that focus on those critical examples). An example of such a combination is the GALE optimizer that intelligently explores thousands of scenarios by examining just a few dozen of the most informative examples. GALE-style reasoning enables a very fast, very wide-ranging exploration of behaviors, as well as the effects of those behaviors' limitations.**

This paper applies GALE to the CDA (continuous descent approach) model within the Georgia Tech WMV (Work Models that Compute) framework. CDA is a model of pilot interactions: with each other and also with the navigation systems critical to safe flight. We show that, using CDA+GALE, it is possible to identify and mitigate factors that make pilots unable to complete all their required tasks in the context of different (1) function allocation strategies, (2) pilot cognitive control strategies, and (3) operational contexts that impact and safe aircraft operation. We also show that other optimization methods can be so slow to run that, without GALE, it might be impractical to find those mitigations.

***Index Terms*—Human Factors, Cognitive Modeling, Multi-objective Optimization, Active Learning**

I. Introduction

There are many advantages of a model-based approach to human factors. Traditional human-in-the-loop (HITL) experimental case-studies are expensive, time consuming, and difficult to reproduce. Model-based conclusions, on the other hand, are reproducible and verifiable (just run the model again). Another advantage of the model-based approach is that models can simulate abstracted real world behavior much faster than real-time; thereby enabling an extensive evaluation of more options than would be possible with HITLs.

In theory, complex models can be analyzed via multi-objective optimizers by running them across many central processing units (CPUs). In practice, that CPU may not be available. For example one of us regularly analyzes a model that needs 30 weeks of CPU time. For high priority issues in need of urgent resolution, then this 30 weeks of computer time can be achieved in five days of parallel execution on NASA's supercomputers. However, in practice researchers can usually access a small fraction of that CPU.

Joseph Krall is a chief data scientist at LoadIQ, Reno, Nevada; e-mail: kralljoe@gmail.com

Tim Menzies is with the Computer Science department, NcState University; email: tim.menzies@gmail.com.

Misty Davies is with the Intelligent Systems Division, NASA Ames Research Center, CA, USA; e-mail: misty.d.davies@nasa.gov.

To address this problem in other domains, we have proposed an optimization method, called GALE [1]–[3], that focuses on a small number of most informative examples. Hence, GALE explores only a few dozen examples rather than the thousands (or more) used by traditional methods [1], [3]. This simplifies and improves our ability to reason about complex cognitive models.

In practice, GALE runs much faster than traditional optimizers. Standard optimization algorithms such as NSGA-II [4] require 3000 to 5000 evaluations to explore the pilot simulator that is the focus of this paper. GALE, on the other hand, performs the same task using 25 to 50 evaluations [2]. In practice, this has tremendous practical implications. When generating conclusions from a randomizing optimizer such as GALE or NSGA-II, it is important to check that the conclusions hold in multiple repeats (say, 20 repeats). This number of evaluations is a critical indicator of runtime in optimizers when the model is complex. For example, a study of the pilot simulator (replicated 20 times) takes 1.5 and 100 hours for GALE and NSGA-II, respectively [2].

The algorithms behind GALE have been presented previously [1]–[3]. Those prior reports focused on runtimes and did not explore the analysis implications of GALE. The core contribution of this paper is a case study on how GALE-style reasoning assists in the analysis of work to support the design of complex operational concepts involving human operators and automated systems. This paper considers a simulation framework designed to investigate function allocation schemes early in the operational concept design process (CDA [5]–[9]) and explores in detail how GALEs conclusions relate to the effect of function allocation strategy, pilot cognitive control strategy, and operational context impact and safe aircraft operation within the context of that model. As shown in §III, (1) GALE offers novel insights into complex cognitive models; (2) other methods would be so slow to run that it might be impractical to find those insights without GALE.

The rest of this paper is structured as follows. An effective search of the models requires the utilization of automated tools such as the multi-objective objective evolutionary algorithms (MOEAs) discussed in §III. These models are intricate and take time to run. One such MOEA is our GALE tool discussed in §III-B which, in §IV is applied to a simulation of pilots flying an automated aircraft under different function allocations. We show that GALE can provide insight into research questions like:

- RQ1: Given a limited maximum human task load capac-

ity, what are the effects to safety assurance when a pilot's workload exceeds his or her capacity?
- RQ2: What are the effects of changing how much pilots rely on automation?
- RQ3: What are the effects of changing pilot policies for monitoring and overlooking flight procedures?

II. Motivation

Mitigation for human-automation interaction errors on the flight deck consists primarily of defining concepts of operation in which the roles and responsibilities of the pilots and of the automation are clearly laid out. As a result, there is research on using models of human-machine interaction in conjunction with technologies such as model-checking [10]–[12] and simulation [5]–[8], [13], [14] at the time of automation design. For example, researchers at Georgia Tech [5]–[8], [13], [14] have developed the WMC (Work Models that Compute) framework.

A. Work Models that Compute

Work Models that Compute (WMC) is a simulation framework, implemented in C++, for modeling and simulating concepts of operation that involve different function allocations between human operators and their automation. WMC work models are representations of the work a team of human and automated agents need to perform to accomplish the objectives of a concept of operation. The representations of work, or work models, are defined by resources and actions (and functions). Resources describe the state of the work environment and actions are the processes by which resources, and thus the work environment, are accessed and modified. Related actions can be aggregated into functions characterizing work serving a specific set of operational objectives.

Agent models in WMC specify the behaviors of an agent in handling taskwork. A basic agent in WMC is able to process actions assigned to it, execute those actions instantaneously, and report the time of action execution. A human performance agent is tailored to be able to address analyzing impacts of human limitations. For example if an action is counted as a unit of capacity for the human agent, the human can get overloaded and may delay actions, or interrupt them for higher priority actions. They may even forget actions altogether.

A human performance agent also may have different degrees of control over a situation. The opportunistic cognitive control mode is defined as "the case in which the next action is chosen from the current context alone and mainly based on the salient features rather than durable goals or intentions" [15]. The agent operating selects the next action in response to assessments that are driven by salient features in the environment. The tactical cognitive control mode is defined as the case in which "the persons event horizon goes beyond the dominant needs of the present, but the possible actions considered are still very much related to the immediate extrapolations from the context" [15, p170]. Agents are guided by rules or patterns of behavior. The strategic cognitive control mode is defined as the one where "the person is using a wider event horizon and looking ahead at higher level goals..." [15]. The agent plans actions.

To investigate function allocation, work is assigned to agents. A specified invocation of work models and agent models comprise a scenario, and an analyst may test different function allocations and agent parameterizations for the same scenario through scripts.

B. Continuous Descent Approach

One concept of operation that has been investigated is optimized descent, also known as continuous descent approach (CDA). WMC's simulation models the physics of flight and the atomic actions of the pilots and the automation, in the context of prioritization schemes.

CDA can understood by contrasting it with another landing tactic, the *standard descent*. In a standard descent, an aircraft must descend via several steps, requesting a new clearance at every step. As a consequence, flight times are longer and more fuel is burned. As the aircraft reroutes and encircles an airport before a runway is clear to land on, these wait times equate to more noise for the city. Additionally, it is harder to fly at lower altitudes due to changes in the atmosphere and wind environments.

By contrast a *continuous descent approach* is a continuous non-stepped descent in which only one request for landing is needed. This simplifies the communication overhead between radio towers and pilots, and avoids extended duration at low altitude. As a result, a continuous descent can more accurately approach the runway, less fuel is burned, and less noise is emitted into the city. Figure 1 illustrates the difference between CDA and traditional landing methods.

III. Learning From Models

For analyzing complex models like the WMC representation of CDA, we prefer GALE to traditional optimizers since those optimizers require certain simplification assumptions. For example, if models are simple continuous equations, then they could be readily explored with gradient descent methods such as the Quasi-Newton method (perhaps using the BFGS update rule recommended by Sims [16]). However, all such gradient descent methods assume that the model being explored is essentially continuous. Models like CDA are not continuous since their internal state space is divided into one combination of each branch of each *if statement* in the code [17].

A better approach for exploring complex models is a multi-objective evolutionary algorithm (MOEA). There are many such optimizers including GALE and more traditional tools such as NSGA-II [4]. MOEAs assume that the model is a function that converts *decisions* "d" into *objective* scores "o"; i.e.

$$o = model(d)$$

In this framework, each pair (d, o) is an *individual* within a *population*. MOEAs try to find a range of good inputs by progressively improving the population using the *generational* approach of Figure 2. Note that MOEAs make no assumptions about model continuity (e.g. unlike the Quasi-Newton methods, they do not assume that models have local smooth gradients). They can also explore trade-offs between goals (see the domination predicate discussed in Figure 2). As discussed below, MOEAs have problems with *brittleness* and *CPU*.

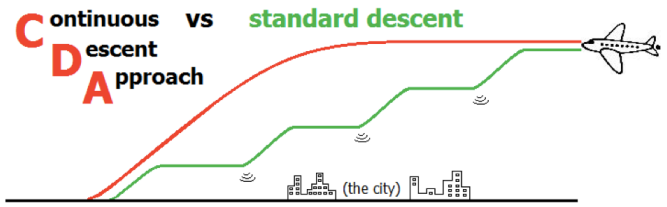

Fig. 1: The red line shows a Continuous Descent Approach (CDA). The green stepped line represents the traditional approach, which is closer to the city, so the city hears more noise emitted from the aircraft.

.

A. Brittleness

Ideally, any insight we glean from a model is not "brittle"; i.e. it is a conclusion that is robust in the face of minor changes to model inputs. Unfortunately, experts in MOEA reasoning caution that many MOEAs generate "brittle" decisions.

According to Harman [19], understanding the neighborhood around individual solutions is an open and pressing issue:

> "It may be better to locate an area of the search space that is rich in fit solutions, rather than identifying an even better solution that is surrounded by a set of far less fit solutions." [20].

He argues that many software model problems are *over-constrained*; i.e. no precise solution over all variables is achievable. Such over-constrained problems are usually explored using heuristic search methods such as the MOEA of Figure 2. The results of such partial heuristic search may be "brittle"; i.e., small changes to the search results may dramatically alter the effectiveness of the solution [20]. One way to check for brittleness is to use *neighborhood perturbation*:

1) *Cluster* a population into local neighborhoods;
2) Build a new population by *perturb*ing the decisions in each neighborhood;
3) Halt if objectives do not change after perturbation;
4) Else, go to step 1.

One reason we endorse the GALE algorithm for reaching conclusions from complex cognitive models is that it directly implements neighborhood perturbation.

An evolutionary multi-objective optimization algorithm (MOEA) requires at least two operators: *cull* and *perturb*: MOEAs generate an initial population by randomly selecting decisions then *culling* the individuals with the lower objective scores. A new population P_n is generated by *perturbing* the decisions of the surviving individuals (e.g. via random mutation or grafting together parts of the decisions of different individuals). MOEA's halt when P_n scores no better than prior generations $P_{m<n}$.
One way to implement the culling (step 2) is via *domination*; i.e. remove one example if it can be shown that it is worse than (a.k.a. "is dominated by") some other examples. Two forms of domination are *binary* and *continuous* domination. In *binary domination*, one individual x dominates y if all of x's objectives are never worse than the objectives in y but at least one objective in solution x is better than its counterpart in y; i.e.

$$\{\forall o_j \in \textit{objectives} \mid \neg(o_{j,x} \prec o_{j,y})\}$$
$$\{\exists o_j \in \textit{objectives} \mid o_{j,x} \succ y_{j,y}\}$$

where $(\prec, \succ)$ tests if an objective score in one individual is (worse,better) than in the other individual.
An alternate culling method is the *continuous domination* predicate [18] that favors y over x if x "losses" least:

$$\begin{array}{rcl} \textit{worse}(x,y) & = & \textit{loss}(x,y) > \textit{loss}(y,x) \\ \textit{loss}(x,y) & = & \sum_j^n -e^{\Delta(j,x,y,n)}/n \\ \Delta(j,x,y,n) & = & w_j(o_{j,x} - o_{j,y})/n \end{array} \quad (1)$$

where "n" is the number of objectives and $w_j \in \{-1, 1\}$ depending on whether we seek to maximize goal x_j.

Fig. 2: Multi-objective evolutionary algorithms.

- NSGA-II [4] uses a non-dominating sort procedure to divide the solutions into *bands* where $band_i$ dominates all of the solutions in $band_{j>i}$ (and NSGA-II favors the least-crowded solutions in the better bands).
- *SPEA2*: favors solutions that dominate the most number of other solutions that are not nearby (to break ties, it uses density sampling) [21];
- *IBEA*: uses continuous dominance to find the solutions that dominate all others [18];
- In *Particle swarm optimization* (PSO), a *particle*'s velocity is 'pulled' towards the individual and the community's best current solution [22];
- The *many-objective optimizers* are designed for very large numbers of objectives [23];
- Multi-objective *differential evolution* (DE): members of the frontier compete (and are possibly replaced) by candidates generated by extrapolation from any three other members of the frontier [24], [25];
- The *decomposition methods* that *first* divide the space of candidate solutions into numerous small regions; then *second* run a relatively simple optimizer in each region [26], [27].

Fig. 3: Some sample MOEAs.

B. Problems with CPU

CDA is a complex model with many input parameters. The input parameter space for such models tends to grow very large so there is a pressing and urgent need for efficient modeling techniques.

The primary design criteria for standard MOEAs is "ability to explore complex trade-offs" and not runtime speed. While the internal details of standard MOEAS may be very different (see Figure 3); most of them share one key characteristic: they evaluate $O(2N)$ examples (twice the population size because they generate offspring which are perturbations of their parent examples) for each generation. One reason we advocate GALE is that it replaces $O(2N)$ with a much faster $O(log_2 N)$ technique (see below).

GALE combines (a) the neighborhood perturbation (described above) with (b) the MOEA algorithm of Figure 2. The algorithm reflects over a *population* of points, each of which contains *decisions* (some inputs to a model). It then searches for the input decisions that lead to best outcomes. For example:

- if we adjust the inputs to the model for (say) high tailwind conditions...

GALE initially builds a population of points by selecting decisions at random. It then *clusters* those decisions into neighborhoods as follows:

1) Find two distant points in that population; call them the *east* and *west* poles.
2) Draw an axis of length c between the poles.
3) Let each point be at distance a, b to the *east, west* poles. Using the cosine rule, project each point onto the axis at $x = (a^2 + c^2 - b^2)/(2c)$.
4) Using the median x value, divide the population.
5) For each half that is larger than $\sqrt{N}$ of the original population, go to step 1.

Note that the above requires a distance measure between sets of decisions: GALE uses the standard case-based reasoning measure defined by Aha et al. [28]. Note also that GALE implements step 1 via the FASTMAP [29] linear-time heuristic:

- Pick any point at random;
- Let *east* be the point furthest from that point;
- Let *west* be the point furthest from *east*.

These final sub-divisions found by this process are the *neighborhoods* that GALE will *perturb* as follows:

- Find the objective scores of the *east, west* poles in each neighborhood.
- Using the continuous domination predicate of Figure 2, find the *better* pole.
- Perturb all points in that neighborhood by pushing them towards the better pole, by a distance $c/2$ (recall that c is the distance between the poles).
- Let generation $i+1$ be the combination of all pushed points from all neighborhoods.

From a formal perspective, GALE is an active learner [30] that builds a piecewise linear approximation to the Pareto frontier [31]. For each piece, it then pushes the neighborhood up the local gradient. This approximation is built in the reduced dimensional space found by the FASTMAP Nyström approximation to the first component of PCA [32].

Fig. 4: Inside GALE

- ... then GALE can report the best monitoring policies for the cockpit instrumentation.

Figure 4 lists the procedure by which GALE clusters the data into neighborhoods, then perturbs each neighborhood. In terms of monitoring for brittleness, the key point of GALE is that this process continues until the perturbations stop having any new effect (i.e. they stop generating better objective scores). That is, all GALE solutions are guaranteed not to be brittle.

Also, in terms of reducing runtime, the key feature of GALE is that, unlike traditional MOEAs such as NSGA-II [4], GALE does not evaluate its entire population. Instead, as it recursively clusters the data in two (using steps 1,2,3,4,5 in Figure 4), GALE only computes the objective scores for the two most distant points in each division. This means that this binary division of the data terminates after just $log_2(N)$ comparisons of evaluated individuals. This is much less than the $2N$ evaluations required by traditional methods like NSGA-II.

Note that the above is a very brief description on GALE. For a full description including algorithms, download sites, and results from dozens of models, see [2], [3].

IV. A Case Study: CDA and GALE

This case study was designed after reflecting on the following. Sometimes, when monitoring cockpit instruments, a pilot is unable to complete all required tasks, which can have adverse consequences on aviation safety. It can occur for many reasons, including fatigue, or unexpected or stressful situations.

A. Goals

The goals of this study are:

1) To automatically discover the implications of lack-of-attention within WMC's CDA model;
2) To learn what mitigations exist (if any) for reducing the problems associated with this lack-of-attention.

We use interrupted, forgotten, and delayed tasks within the CDA model as an incomplete proxy for a safety metric. Our possible mitigations are restricted to the input of our CDA problem space.

B. Methods

1) Apparatus: The CDA work model implemented in WMC is a model of pilot interactions. CDA employs a continuous descent approach to a runway. As aforementioned, a continuous descent is an alternative to the standard approach to a runway. A continuous descent approach is arguably much more efficient in terms of a) fuel economy and cost, b) noise, and c) flight duration.

CDA is packaged within the WMC (Work Models that Compute) suite. As mentioned, human performance agents in WMC have a "maximum human task load" variable which describes how much they can handle. If the task load is too large, then some of the workload will not be completed in time, or worse yet, might go forgotten entirely. We use the metrics that describe these problems as the output objectives for GALE, i.e., the dependent variables which are detailed in the subsection just below.

Also as mentioned, human performance agents in WMC may employ different cognitive control strategies to alleviate the problems of large task loads. By studying the different strategies in terms of the output objective scores (the dependent variables), it is possible to understand the safety implications of different strategies and to learn what mitigations might exist (and can be exploited for safety gain).

2) Independent Variables: A CDA "problem instance" defined within the WMC framework consists of four decisions and five objectives. The CDA implementation itself within WMC contains many other inputs (for example, the flight path and aircraft type are fixed) but for the purposes of this study, they are held constant.

a) Maximum human task load (HTM). This value describes how many tasks (where a task is an atomic action) can be maintained in a mental to-do list by a person. Tasks in the model are assigned a duration and a priority [5]. When the number of necessary tasks exceeds the number of tasks that the person can maintain, there can be incurred delays, errors, or the possibility of the task being forgotten and lost.

b) Function allocation (FA). This variable refers mainly to the relative authority between the human pilot and the avionics. FA defines the different ways the pilots can configure the autoflight controls.

Highly Automated: The computer processes most of the flight instructions directly; the pilot only confirms the clearances. This function allocation addresses a possible future concept of operation.

Mostly Automated: The pilot processes the instructions and programs the autoflight system, but then the autoflight system controls the flight path automatically. This mimics current lateral "LNAV" and vertical "VNAV" flight navigation operations.

Mixed-Automated: The pilot processes the instructions and programs the computer to handle the lateral flight path. The flight crew uses a lower level of automation for the vertical profile, including the altitude, vertical speed, and airspeed.

c) Contextual control mode (CCM). These describe the pilots' ability to apply patterns of activity in response to the demands and resources in the environment. These are based on Hollnagel's work on representative patterns of activity [15]:

Opportunistic: Pilots monitor and perform tasks related to only the most critical functions (monitoring is conducted only when necessary to complete assigned taskwork).

Tactical: Pilots cycle through most of the available monitoring tasks, and double check some of the computer's tasks (periodic monitoring checks are conducted).

Strategic: Pilots cycle through all of the available monitoring tasks, and try to anticipate future tasks.

Scenario (SC). WMCs CDA model includes four different arrival and approach scenarios. The four arrival and approach scenarios (SC) implemented within the CDA model are:

Nominal: (ideal) arrival and approach.

Late Descent: controller delays the initial descent.

Unpredicted rerouting: pilots directed to an unexpected waypoint.

Tailwind: wind pushes plane from ideal trajectory.

3) Dependent Variables: CDA's five objectives keep track of how many tasks were delayed, interrupted or forgotten entirely (which impacts the relative safety of the flight itself) [5]. Better pilot organizational structures can be found by exploring different inputs of CDA to optimize these goals so that safety is improved:

Num Forgotten Actions: tasks forgotten by the pilot. When the number of tasks expected of the pilot exceed the HTM, tasks with the lowest priority are 'forgotten'.

Num Delayed Actions: number of delayed actions. Tasks have a scheduled time and a duration. When a higher priority task causes another task to begin later than its scheduled time, it is 'delayed'.

Num Interrupted Actions: interrupted actions. A higher priority task can cause a lower priority task to be interrupted.

Interrupted Time: time spent on interruptions.

Delayed Time: total time of all of the delays.

4) Experimental Design: In the following, CDA was run for varying and decreasing values of HTM. For each HTM value, we collect:

- *The baseline objective scores* seen in CDA.
- *The treated objective scores* seen in CDA. These treatments are learned by GALE and represent the best case actions that could be performed by pilots to mitigate against the lack-of-attention problem.

When analyzing CDA, GALE was run using parameters found to work best on several other models [2]:

- GALE uses a population of size 100;
- GALE must terminate after a maximum number of 20 generations;
- GALE may terminate if no improvement is seen in any objective in the last three generations;

To control for random effects during optimization, all scores are the mean values of 20 repeated runs of *baseline* or the model *treated* with GALE's conclusions.

5) Data Analysis: We compare *baseline* and *treated* by letting GALE select any inputs across the full range of all CDA input values.

That first experiment found a curious threshold effect: underneath a certain HTM point, all the best decisions concerned a particular contextual control mode. In *Opportunistic* mode, pilots monitored and performed tasks related to only the most critical functions in the cockpit. That is, in this mode, pilots executed only the most essential monitoring actions according to the CDA model (e.g. monitoring altitude and monitoring descent airspeed), and focused primarily on adjusting the lateral profile.

To understand the impact of this *Opportunistic* mode, a followup experiment was conducted, with the *Opportunistic* mode disabled.

6) Sanity Checks: We define two sanity checks on our results:

Sanity check #1: As we *decrease* HTM (maximum human task load), we should see *increasing* adverse flight operations.

Sanity check #2: Using CDA and GALE, we can find mitigations that reduce the effects of decreasing HTM. If this were otherwise, that would suggest that MOEAs like GALE are not useful here.

V. Results

A. Results with Opportunistic CCM

Figure 5 summarizes the results. As an aid to help visualization, bar graphs are added to each column (and the bar with the largest, smallest value shows the max,min values in the column). GALE's decisions are shown at the top. Beneath that, we show two sets of objective scores:

- *The baseline runs* which are the average objective scores seen without using GALE (just from randomly selecting input decisions).
- *The treated runs* which are the average final objective scores seen after 20 runs of GALE.

For the tables of objective results:

- The controlled value (HTM, maximum human task load) is shown on the left hand side of the table.

	Level of Autonomy			Scenario				Contextual Control Mode		
HTM	HIGH	MOST	MIX	NORM	LATE	ROUT	WIND	OPP	TAC	STR
1	88%	12%	0%	31%	31%	24%	14%	100%	0%	0%
2	14%	86%	0%	16%	12%	28%	44%	96%	4%	0%
3	12%	84%	4%	26%	12%	23%	39%	72%	26%	2%
4	33%	57%	10%	33%	12%	31%	24%	14%	84%	2%
5	41%	57%	2%	39%	16%	26%	18%	8%	87%	5%
6	21%	79%	0%	31%	15%	44%	10%	2%	98%	0%
7	46%	52%	2%	48%	14%	21%	16%	4%	91%	5%
8	32%	68%	0%	30%	20%	38%	13%	4%	91%	5%

HIGH = Highly Automated.
MOST = Mostly Automated.
MIX = Mixed Auto/Manual

NORM = Nominal Descent
LATE = Late Descent Scenario
ROUT = Unexpected Rerouting
WIND = Unexpected Tailwind

OPP = Opportunistic
TAC = Tactical
STR = Strategical

	HTM = Max Human Taskload	nFA= num Forgotten Actions (per minute)	nDA = num Delayed Actions (per minute)	nIA = num Interupted Actions (per minute)	AIT = avg Interupted Time (secs per minute)	aDT = avg Delayed Time (secs per minute)
baseline (no optimizing)	1	1087	34	9	60	21
	2	752	11	5	13	7
	3	466	5	3	3	3
	4	270	2	2	1	2
	5	151	1	2	0	1
	6	99	0	1	0	1
	7	57	0	1	0	1
	8	18	0	0	0	0
GALE (optimizing)	1	54	2	0	8	0
	2	39	0	0	1	1
	3	46	0	0	0	1
	4	140	1	1	1	1
	5	108	1	1	0	1
	6	61	0	1	0	1
	7	21	0	1	0	1
	8	1	0	0	0	0

Fig. 5: All contextual control modes enabled. Forgotten, delayed, and interrupted actions are reported by the simulation at each 0.05s time step. a) Decisions found by GALE; b) Objectives obtained.

- The values in the other columns are all counts per simulated minute.

When reading these results, it is insightful to look for *saturation*, *trends*, *absences* and *cliffs*.

1) Saturation: Values *saturate* when they are driven towards their theoretical maximum. In the case of the CDA objectives, any *Time* objective that occurs at 60 seconds per minute is *saturated*. Such saturation can be observed in the *avg Interrupted Time* and the *avg Delayed Time* values of 60 at $HTM = 1$ in Figure 5b .

In terms of a pilot maintaining safe operations, saturation is highly undesirable. At saturation, a pilot is permanently interrupted for all tasks so everything gets delayed. Note that this saturation result satisfies one of our *sanity checks* (that less HTM leads to more problems).

The good news is that saturation can be avoided. In Figure 5b, we see that the simulated pilots following GALE's advice never reach saturation at $HTM = 1$. GALE advises the simulated pilots to restrict themselves to only the most important tasks (in CDA's model, this means operating in *opportunistic* mode) and to allow the automation to handle all or most of the tasks that the automation can handle. Note that this means that our experiment satisfies another *sanity check* (that GALE can learn mitigations to the low HTM problems).

2) Trends: *Trends* are values that change smoothly with changes to HTM. For example, the *num Forgotten Acts* per minute is low in all results until HTM falls below four (after which time it can spike to alarmingly large values).

With one exception, this trend holds for all objectives—which is to say that airplane safety is critically dependent on this HTM value.

The exceptions to this trend are the GALE results of

Figure 5b. In those rows, GALE could learn mitigations that compensate for pilots struggling to control. Those mitigations are shown in Figure 5a.

3) Absence: Several columns in Figure 5a contain nearly all zero values. That is they are mostly *absent* from the recommendations made by GALE.

- A *MIX*ed level of autonomy was rarely useful, suggesting that the simulated pilots should very rarely program the computer to handle only some of the airplane instructions.
- Similarly, the *STR* strategic cognitive control mode was also rarely used. From this result, we say that, in this model, pilots should avoid cycling through all of the available monitoring tasks while trying to anticipate future tasks.
- Other absent columns can be seen in the scenarios GALE found it could handle. GALE's recommendations to the simulated pilot could handle all the *Scenario*s (see all the non-zero numbers in the *Scenario* columns of Figure 5b).

4) Cliffs: Cliffs are values that change sharply between one HTM value and the next; there are two large cliffs in Figure 5.

The first cliff relates to *Level of Autonomy*. At $HTM = 1$, GALE nearly always selected for a *HIGH* level of autonomy. However, as soon as pilots can do or hold in memory two things at once (at $HTM > 1$), that recommendation no longer holds. In fact, for all levels of $HTM > 1$, GALE usually prefers for the pilot to process flight data and program the computer (i.e. to use the *MOST* approach). Another way to say this is that, when their attention is failing, our simulated human pilots should give more tasks to the machines. However, at any other time, it is better to guide, and not be guided by, the automatic systems.

The other cliff in these results is seen in the right-hand-side columns of Figure 5a. In those results, it can be seen that for $1 \leq HTM \leq 3$ the *OPP* (opportunistic) cognitive control mode is most often selected by GALE. However, above that point (for $HTM > 3$), it is rarely selected. This cliff is a large enough effect in a critical range of the model to deserve special attention. Accordingly, Experiment #2 (discussed below) explores the the relative merits of opportunistic control versus other modes.

B. Results with Dsiabling Opportunistic CCM

As shown in Figure 5a, at low HTM levels of $HTM < 4$, most of GALE's recommended actions use the opportunistic cognitive control mode (where pilots monitor and perform tasks related to only the most critical functions). To see if this was an important effect, we repeated the above experiment with this opportunistic mode disabled.

The results are shown in Figure 6. In those results, the following aspects are noteworthy:

- Comparing the *baseline* and *GALE* distributions of Figure 6b, we see that the GALE treatment barely changes the baseline distributions. That is, if GALE cannot use the opportunistic control mode, then it cannot mitigate for low HTM values.
- Comparing the *Scenario* results in Figure 6a to Figure 5a, we see that when we cannot use opportunistic mode there are more absent columns in *LATE* and *WIND*. That is, if GALE cannot use opportunistic control, then it cannot find another mitigation for late arrival or high wind conditions.

From these results, we say that opportunistic mode is an essential tool for combating the problems associated with low HTM in the CDA model.

VI. Discussion

In principle, all the above conclusions could have been reached using an MOEA like NSGA-II. However, in practice, that would have been impractically slow. To understand why, we must review the systems-level tasks associated with conducting this kind of study. *Commissioning* this CDA model took several months as one of us (Krall) worked inside the NASA firewalls to port the CDA model to the NASA servers. In that process, CDA was run many times to "iron out the bugs". Often it was necessary to trace through the evaluations to determine what was going astray. During this period, we were grateful that GALE was only making $O(log(N))$ evaluations per generation since the $O(2N)$ evaluations used by standard optimizers would have led to an overwhelming amount of data.

Also note that after commissioning the model came *generating conclusions*. This required 20 repeats of all models for baselines and with GALE, repeated for HTM set from one to eight, then repeated twice with and without the opportunistic CCM. With GALE, those runs took 83 hours and with NSGA-II, those runs would have taken much longer. Based on some samples we made of NSGA-II performing parts of running with the opportunistic CCM, we estimate that if NSGA-II was used for the above experiments, then that would have taken 6 months.

As to the external validity of this work, all the conclusions made here came from two tools: the CDA model and GALE. If these tools are somehow distorted or biased then our conclusions would be distorted or biased in the same manner. That said, there is enough prior work on CDA and GALE to make the case that it useful to study the CDA model with the GALE optimizer. We note that these tools are the products of years of research and much analysis and testing [1]–[3], [5]–[9], [13], [14]. Given the resources spent on its construction, it seems prudent and timely to learn what we can from that model.

VII. Conclusion

A common, and naive, assumption made by researchers who have not conducted model-based experiments is this: once the model is built, then inference is easy.

We have shown in this paper that, for large and complex models, this naive assumption may not hold. In fact, it is critically important to consider *how* that inference is conducted. This paper endorses the use of modeling for complex studies, but it also addresses a few issues that must also be handled whenever learners are used in conjunction with models.

Firstly, there is a need to develop and commission the model for integration with the learner tools. This can be a time-intensive task and moreover, additional modifications can be

	Level of Autonomy			Scenario				Contextual Control Mode		
HTM	HIGH	MOST	MIX	NORM	LATE	ROUT	WIND	OPP	TAC	STR
1	66%	32%	2%	35%	21%	17%	26%	x	94%	6%
2	87%	13%	0%	48%	0%	21%	31%	x	94%	6%
3	43%	55%	2%	78%	0%	20%	2%	x	96%	4%
4	40%	60%	0%	73%	0%	21%	6%	x	100%	0%
5	67%	33%	0%	81%	2%	18%	0%	x	95%	5%
6	27%	73%	0%	80%	2%	18%	0%	x	100%	0%
7	38%	60%	2%	83%	2%	15%	0%	x	98%	2%
8	43%	57%	0%	81%	0%	19%	0%	x	98%	2%

HIGH = Highly Automated.
MOST = Mostly Automated.
MIX = Mixed Auto/Manual

NORM = Nominal Descent
LATE = Late Descent Scenario
ROUT = Unexpected Rerouting
WIND = Unexpected Tailwind

OPP = Opportunistic
TAC = Tactical
STR = Strategical

	HTM = Max Human Taskload	nFA= num Forgotten Actions (per minute)	nDA = num Delayed Actions (per minute)	nIA = num Interupted Actions (per minute)	AIT = avg Interupted Time (secs per minute)	aDT = avg Delayed Time (secs per minute)
baseline (no optimizing)	1	1333	44	11	60	25
	2	958	14	6	15	8
	3	562	6	4	4	4
	4	295	2	2	1	2
	5	151	1	2	0	1
	6	100	0	1	0	1
	7	60	0	1	0	1
	8	19	0	0	0	0
GALE (optimizing)	1	1389	38	11	60	27
	2	684	10	5	8	7
	3	394	4	3	2	3
	4	189	2	2	1	1
	5	102	1	1	0	1
	6	56	0	1	0	1
	7	19	0	1	0	1
	8	1	0	0	0	0

Fig. 6: Opportunistic mode disabled. Forgotten, delayed, and interrupted actions are reported by the simulation at each 0.05s time step. a) Decisions found by GALE; b) Objectives obtained.

cause for restarting the actual experimentation which follows thereafter.

Secondly, the learners themselves are complex computational intelligence software tools which have been the subject of decades of research. The selection and deployment of just a single tool can become a complex decision process. Moreover, high model runtimes can restrain the space of usable learning tools by a vast amount as much of the research on those learners has focused on optimizing very small models.

In this paper, GALE was used because it can optimize very large models. This sort of enabling technology is made possible because GALE does not need to run the models as often as other learning tools (specifically, GALE performs $O(log(N))$ evaluations while standard methods explore a space of $O(2N)$ options). GALE can quickly generate conclusions from complex models. For example, in the case study explored here we offer the following answers to the research questions posed in the introduction.

A. RQ1: Safety and Low HTM

We say that unsafe operation occurs if pilots cannot complete their assigned tasks. For little to no cognitive limitations (HTM is 4 or higher), there is very little effect on taskload completion times. The reason for this is simple: if the pilot can perform all tasks as they come, then there should not be any backlog of delayed tasks. For lower levels of HTM, there were many delays and interruptions noted for our simulated pilots. Hence, we conclude that low HTM levels are especially dangerous within this model.

B. RQ2: Impact of Automation

For nearly all levels of HTM, it was sufficient to rely on a *MOST* level of autonomy where the pilot is in charge

of processing input and programming the cockpit computers. However, in extreme situations ($HTM = 1$), that recommendation is not supported. For such very low levels of HTM, our simulated pilots should switch to *HIGH* levels of automation to ensure aircraft safety.

This recommendation intuitively makes sense if we assume that the automation works as the pilot intends and that the pilot understands how it works. We note that the creators of WMC have published a study with a version of their model tuned to study automation surprise [12].

C. RQ3: Pilot Monitoring Policies

As to appropriate cognitive control modes for watching over an aircraft, for higher levels of HTM, it was sufficient to step up to the tactical control mode (which allows the pilot to monitor more of the aircraft flight procedures). For lower levels of HTM, tactical flight operations proved to be too much for our simulated pilot to handle and opportunistic control mode was essential to mitigate against low HTM.

More validation should be done before applying the recommendations we make here for the CDA model to the real-world safety problems that have inspired both CDA and this study. Accidents such as Air France 447 [33] and Asiana Flight 214 [34] show that the pilots became distracted by off-nominal behavior, and *failed to monitor* the most important flight state information (e.g. airspeed, altitude and attitude). When stressed, pilots are asked to do something very like the opportunistic mode as implemented within the CDA model—worry about the key monitoring and flight tasks first.

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 partially 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

[1] J. Krall, T. Menzies, and M. Davies, "Learning the task management space of an aircraft approach model," in *Proceedings of the 2014 AAAI Workshop*, ser. AAAI'14, 2014.

[2] J. Krall, "Faster evolutionary multi-objective optimization via GALE: the geometric active learner," Ph.D. dissertation, West Virginia University, 2014.

[3] J. Krall, T. Menzies, and M. Davies, "Geometric active learning for software engineering," *Under-Review, IEEE TSE*, 2014.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2000.

[5] S. Y. Kim, "Model-based metrics of human-automation function allocation in complex work environments," Ph.D. dissertation, Georgia Institute of Technology, 2011.

[6] A. R. Pritchett, H. C. Christmann, and M. S. Bigelow, "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*, Miami Beach, FL, 2011.

[7] K. M. Feigh, M. C. Dorneich, and C. C. Hayes, "Toward a characterization of adaptive systems: A framework for researchers and system designers," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 54, no. 6, pp. 1008–1024, 2012.

[8] S. Y. Kim, A. R. Pritchett, and K. M. Feigh, "Measuring human-automation function allocation," *Journal of Cognitive Engineering and Decision Making*, vol. 8, no. 1, 2014.

[9] A. R. Pritchett, S. Y. Kim, and K. M. Feigh, "Modeling human-automation function allocation," *Journal of Cognitive Engineering and Decision Making*, vol. 8, no. 1, 2014.

[10] M. Bolton, R. Siminiceanu, and E. Bass, "A systematic approach to model checking human-automation interaction using task-analytic models." *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 5, pp. 961–976, 2011.

[11] N. Rungta, G. Brat, W. Clancey, C. Linde, F. Raimondi, S. Chin, and M. Shafto, "Aviation safety: Modeling and analyzing complex interactions between humans and automated systems," in *International Conference on Application and Theory of Automation in Command and Control Systems (ATACCS)*, May 2013.

[12] G. Gelman, K. M. Feigh, and J. Rushby, "Example of a complementary use of model checking and human performance simulation," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 5, Oct. 2014.

[13] A. R. Pritchett, K. M. Feigh, S. Y. Kim., and S. K. Kannan, "Work models that compute to describe multiagent concepts of operation: Part 1." *Journal of Aerospace Information Systems*, vol. 11, no. 10, Oct. 2014.

[14] K. M. Feigh, A. R. Pritchett, S. Mamessier, and G. Gelman, "Generic agent models for simulations of concepts of operation: Part 2," *Journal of Aerospace Information Systems*, vol. 11, no. 10, Oct. 2014.

[15] E. Hollnagel, *Human Reliability Analysis: Context and Control*. London: Academic Press, 1993, pp. 159–202.

[16] C. Sims, "Matlab optimization software," 1999.

[17] M. Davies, C. Pasareanu, and V. Raman, "Symbolic execution enhanced system testing," in *Verified Software: Theories, Tools, Experiments*. Springer Berlin Heidelberg, 2012, pp. 294–309.

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

[19] M. Harman and B. Jones, "Search-based software engineering," *Journal of Information and Software Technology*, vol. 43, pp. 833–839, December 2001.

[20] M. Harman and J. Wegener, "Getting results from search-based approaches to software engineering," in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 728–729.

[21] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation, and Control*. CIMNE, Barcelona, Spain, 2002, pp. 95–100.

[22] H. Pan, M. Zheng, and X. Han, "Particle swarm-simulated annealing fusion algorithm and its application in function optimization," in *International Conference on Computer Science and Software Engineering*, 2008, pp. 78–81.

[23] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 577–601, Aug 2014.

[24] R. Storn and K. Price, "Differential evolution– a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[25] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, Feb 2011.

[26] K. Deb, M. Mohan, and S. Mishra, "Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions." *Evolutionary Computation*, vol. 13, no. 4, pp. 501–525, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/ec/ec13.html#DebMM05

[27] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *Trans. Evol. Comp*, vol. 11, no. 6, pp. 712–731, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2007.892759

[28] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, January 1991.

[29] C. Faloutsos and K.-I. Lin, "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*, 1995, pp. 163–174.

[30] S. Dasgupta, "Analysis of a greedy active learning strategy," *in Neural Information Processing Systems 17:*, vol. 1, no. x, 2005.

[31] M. Zuluaga, A. Krause, G. Sergent, and M. Püschel, "Active learning for multi-objective optimization," in *International Conference on Machine Learning (ICML)*, 2013.
[32] J. C. Platt, "FastMap, MetricMap, and Landmark MDS are all Nyström algorithms," in *In Proceedings of 10th International Workshop on Artificial Intelligence and Statistics*, 2005, pp. 261–268.
[33] "Final report on the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro - Paris," BEA, Tech. Rep., 2012.
[34] "Descent below visual glidepath and impact with seawall asiana airlines flight 214 boeing 777-200er, hl7742 san francisco, california july 6, 2013," National Transportation Safety Board, Tech. Rep., 2014.

Joseph Krall (Ph.D., WVU) is J chief data scientist at LoadIQ, Reno, Nevada, a high-tech start-up that researches and investigates cheaper energy solutions. His research relates to the application of intelligent machine learning and data mining algorithms to solve NP-Hard classification problems. Further research interests lie with multi-objective evolutionary algorithms, search based software engineering, games studies, game development, artificial intelligence, and data mining.

Tim Menzies (Ph.D., UNSW) is a Professor in CS at NcState His research interests include software engineering (SE), data mining, artificial intelligence, and search-based SE, open access science.

Misty Davies (Ph.D. Stanford),is a Computer Research Engineer at NASA Ames Research Center, working within the Robust Software Engineering Technical Area. Her work focuses on predicting the behavior of complex, engineered systems early in design as a way to improve their safety, reliability, performance, and cost. Her approach combines nascent ideas within systems theory and within the mathematics of multi-scale physics modeling.