Stress Testing of Unmanned Traffic Management Decision Making Systems

Xuxi Yang* and Peng Wei†
Iowa State University, Ames, IA, 50011, USA

Maxim Egorov‡, Steven Munn§ and Antony Evans¶,
Airbus UTM, San Francisco, CA, 94086, USA

The emergence of new operations, such as package delivery and air taxis, that could co-exist in the current airspace has been a topic of great interest in recent years. A big challenge associated with the introduction of these new operations is ensuring that the complex system-of-systems responsible for these operations known as Unmanned Traffic Management (UTM) meet the high safety standards of aviation. In this work, we present a general purpose framework for validating decision making systems, protocols, and algorithms that may exist in the UTM ecosystem. We propose a novel, simulation driven approach for validation in UTM that can automatically discover the failure modes of a decision making system, and optimize the parameters that configure the system to improve its performance. We apply this approach to Urban Air Mobility (UAM) and package delivery use cases, and examine in detail the failure modes and the potential improvements of two UTM services critical for strategic deconfliction: trajectory planning and optimal scheduling. Using simulation, we demonstrate that our algorithm is able to discover failure modes in the system that would be challenging for humans to find and fix, and we show how the algorithm can learn from these failure modes to improve the performance of the UTM services in question. We also demonstrate the significant sample efficiency improvements that our algorithm has over naive stress testing approaches that rely on uniform sampling.

I. Introduction

From delivery drones to autonomous vertical take-off and landing (VTOL) passenger aircraft, applications of unmanned aircraft systems (UAS) have become a key part of the discussion about the future of our skies [1]. It is estimated that the number of UAS will dramatically increase within the next 20 years [2–4], and new approaches will be needed to manage this influx of aircraft outside of traditional air traffic management (ATM) systems. To accommodate these types of operations, a digitized, and automated solution has emerged known as Unmanned Traffic Management (UTM) [5]. While UTM is a complex system and has many technological challenges that need to be overcome, ensuring that the decision making systems within UTM operate safely and efficiently is one of the biggest.

In the UTM ecosystem a collection of services is responsible fully or in part for coordination, deconfliction, demand-capacity management, and other advanced functions necessary for enabling these new types of operations. While traditionally, many of these advanced functions were overseen by humans [6], in the UTM ecosystem, they may be fully performed by complex autonomous systems. Such services will require a validation process that must ensure that these systems meet the appropriate safety and reliability criteria. However, validating the safety of autonomous systems can be challenging, because these systems are inherently complex internally, and interact with external systems and environments in non-trivial ways. One way of validating the safety of an autonomous UTM service is to test it with every type of possible input, under every random disturbance and every type of sensor error. However, an exhaustive search over all possibilities that could exist in UTM operations is both intractable computationally and infeasible from an engineering standpoint. Furthermore, system failures can be exceptionally rare, and difficult to reach under nominal conditions, making them hard to identify.

*Graduate Research Assistant, Department of Aerospace Engineering, xuxiyang@iastate.edu. Student Member AIAA.
†Assistant Professor, Department of Aerospace Engineering, pwei@iastate.edu. Senior Member AIAA.
‡Research Scientist, maxim.egorov@airbus-sv.com.
§steven.munn.ctr@airbus-sv.com.
¶Ph.D., tony.evans@airbus-sv.com.

Despite the difficulties of autonomous system validation, a number of computational approaches have been developed to assist in the process. The approaches can be roughly divided into two groups: formal methods based on reachability analysis, and simulation-driven approaches. Formal methods attempt to prove that the safety properties of a system hold by constructing a mathematical model of the system. Methods based on automatic theorem proving (ATP) [7] are used to generate proofs about the safety of the system by modeling the underlying system and its assumptions using mathematical logic. Approaches like probabilistic model checking (PMC) can explicitly model stochastic properties of the system by verifying the system safety properties over Markov models [8, 9]. Hybrid systems proving can capture both discrete and continuous dynamic behavior [10], and is generally more suitable for complex autonomous systems. The drawback of these approaches lies in the modeling limitations the methods can support, and the ambiguity around the fact that if the algorithm is not able to generate a verification it does not prove the system is safe.

Simulation-driven approaches, on the other hand, use a dynamic simulation model to evaluate the performance of a system using a finite number of simulation paths or scenarios. Due to the limited number of constraints on the implementation of simulation models, they are generally more flexible than formal methods for validation and can be used to model environments and systems at multiple levels of fidelity. The simulation scenarios can be hand-crafted by domain experts or by performing parametric sweeps over a low dimensional configuration space [11]. They can also be directly drawn from a stochastic model representing the system's environment.

Simulation driven approaches have recently been applied in practice for validation of autonomous systems used in autonomous driving [12] as well as in aviation [13]. However, the Monte Carlo nature of simulation driven approaches can be inefficient given the large scenario space, rarity of failure events, and the undirected nature of the search. To address this issue, adaptive stress testing has been recently proposed as a practical approach to finding most-likely failure scenarios. The stress testing problem is formulated as a Markov decision process (MDP) and solved by computing or learning a stress testing policy [14–16]. The key idea of these approaches is a carefully designed reward function that leads to the discovery of system's failure modes. However, these approaches can be computationally expensive because they require either learning or computing a stress testing policy, and they are tailored to systems that perform sequential decision making.

In this work, we build upon the adaptive stress testing paradigm, but propose a new methodology for finding the failure modes of UTM decision making systems. To that end, this work makes the following contributions: (1) propose a novel and sample efficient approach for adaptive stress testing that uses Bayesian optimization, (2) expand on the approach in a way that enables the methodology to also optimize the system with respect to the discovered failure modes, (3) apply this approach to validate and optimize strategic UTM decision making services for the UAM and the package delivery uses cases.

II. Optimization Based Adaptive Stress Testing

The adaptive stress testing methodology relies on simulation to discover failure modes of a system under test. This is done by modifying properties of the environment in which the system operates such that the system fails, or by directly guiding the dynamics of the system into a failure mode. The characteristics of the environment or the dynamic path of the system that led to the failure can then be directly examined and addressed.

Consider a system that is being tested by a domain expert in Fig. 1. In this testing environment, the domain expert chooses the input parameters for the simulator, performs the simulation, and examines the output metrics of interest after the simulation is completed. Based on the output metrics, the domain expert may choose to modify the simulation parameters to further test the system. The traditional adaptive stress testing methodology, creates a feedback loop between the metrics of interest generated by the simulation and the parameters input into the simulation. Traditionally, the modified parameter is typically the random seed that configures any random numbers generated by the environment [14], and the system under stress is held fixed.

In this work, we make two critical changes to the adaptive stress testing methodology: (1) we close the feedback loop using an efficient black box optimizer instead of an MDP solver, and (2) we allow the system to modify both the environment parameters and the system parameters (see Fig. 2). These modifications reduce the problem of adaptive stress testing to a task of performing an iterative optimization, where the goal of the optimizer is to discover the simulation scenarios or the simulation paths that lead to failure modes. The objective of the optimization can be constructed by using the output metrics from the simulator that serve as indicators of system failure. For autonomous traffic management systems, metrics tracking near mid air collisions (NMACs) tend to be strong indicators of system failure, and are adopted in this work. The parameter space over which the optimizer is searching is represented by the joint parameter space of the environment and the system under test. The result of the optimization is a set of environment

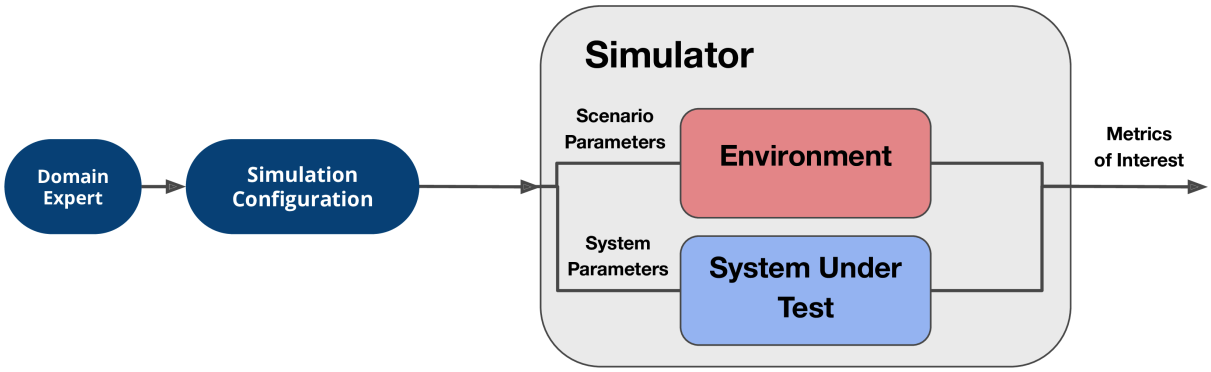


Fig. 1 Simulation-driven testing of a generic system that can be parameterized and evaluated in a configurable environment.

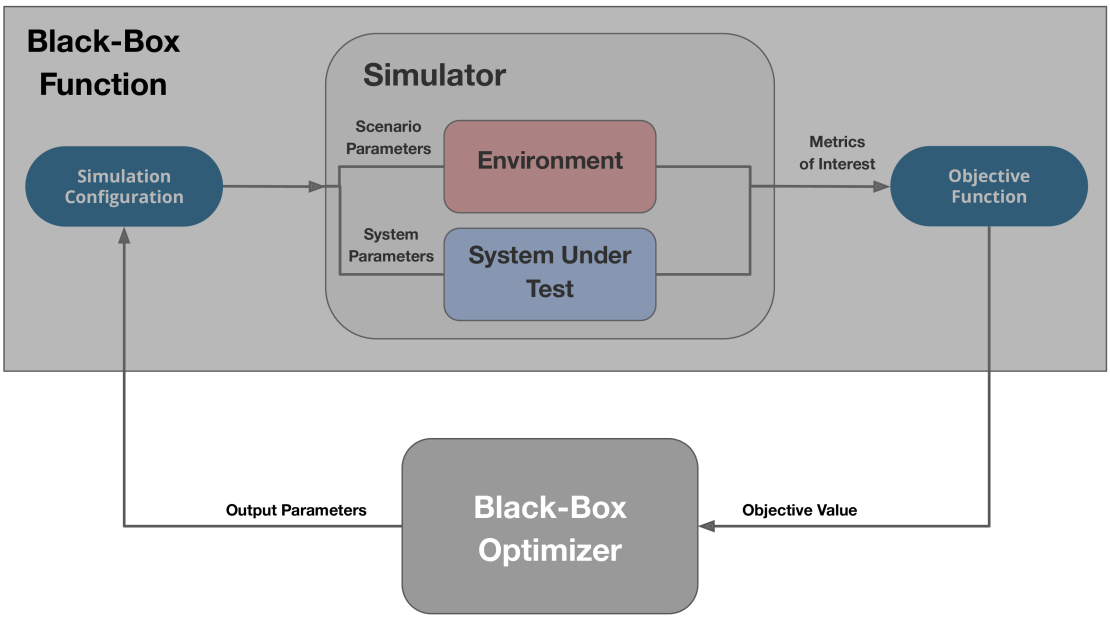


Fig. 2 Adaptive stress testing performed with a black-box optimizer.

scenarios and a corresponding system parameterization that leads to the failure of that instance of the system being tested. For example, consider an adaptive test case attempting to validate an autonomous strategic deconfliction service. The objective is formulated using the frequency of NMACs in the simulation, and the optimization discovers that a package delivery scenario with four intersecting operations leaves one of the intersecting operations in conflict. This scenario configuration ultimately leads to an NMAC in the simulation, which is reflected in the objective, and is further exploited by the stress tester. Upon additional trials, the stress tester discovers that a specific intersection angle between operations in many vehicle conflict scenarios always leads to conflicts. This discovery can be used by subject matter experts to redesign the system in a way that improves safety. We apply this methodology in the remainder of the paper to discover unique and non-trivial failure modes that cover a variety of UTM use cases.

III. Bayesian Optimization with Gaussian Process Priors

Bayesian optimization is a derivative-free sequential design strategy for global optimization of black-box functions [17], which has been shown to outperform other state of the art global optimization algorithms on a number of challenging optimization benchmark functions [18, 19]. For continuous functions, Bayesian optimization typically works by assuming the unknown function was sampled from a Gaussian process (prior) and maintains a posterior distribution for this function as observations are made sequentially. We will briefly overview Gaussian Processes and Bayesian Optimization in the next subsections.

A. Gaussian Processes

Gaussian Process (GP) provides a convenient and powerful class of non-parametric statistical models over function spaces. Specifically, a Gaussian process is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution [20]. In particular, a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ is said to be drawn from a Gaussian process with mean function $m(\cdot)$ and covariance function $k(\cdot,\cdot)$ if for any finite set of elements $x_1, ..., x_m \in \mathcal{X}$, the associated finite set of random variables $f(x_1), ..., f(x_m)$ have distribution

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_m) \end{bmatrix}, \begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_m) \\ \vdots & \ddots & \vdots \\ k(x_m,x_1) & \cdots & k(x_m,x_m) \end{bmatrix}\right) \tag{1}$$

we denote this using the notation

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot)) \tag{2}$$

In this work, we want to use the Gaussian Process to predict the function value for a test point x_*. Suppose we have a training set $X = \{(x_i, f(x_i)|i = 1, ..., n\}$, then the function value f_* at point x_* will have distribution:

$$f_*|x_*, X \sim \mathcal{N}(K(x_*,X)K(X,X)^{-1}f, K(x_*,x_*) - K(x_*,X)K(X,X)^{-1}K(X,x_*)) \tag{3}$$

where f is a vector representing all the function values, and K is the matrix composed of elements representing the covariance function $k(\cdot,\cdot)$. From this equation we can see that the GP provides not just information about the likely value of the function f, but importantly also about the uncertainty around this value.

B. Acquisition Functions for Bayesian Optimization

We assume that the function of interest $f(x)$ is drawn from a Gaussian process prior. Together with the observation data $X = \{(x_i, f(x_i)|i = 1, ..., n\}$ we can induce a posterior over functions. The acquisition function $a(x)$ determines what point in the variable space should be evaluated next via a proxy optimization $x_{next} = \arg\max_x a(x)$, which is our current best guess for the global optima. The following are several popular choices of acquisition function.

Probability of Improvement (PI) is proposed [21] to maximize the probability of improving over the best current value. Alternatively, one could also choose to maximize the Expected Improvement (EI) over the current best. A more recent development is Upper Confidence Bound (UCB) which aims to minimize the regret over the optimization process [22]. The UCB acquisition function has the following form:

$$a_{UCB}(x|X) = \mu(x|X) + \kappa\sigma(x|X) \tag{4}$$

where a tunable κ is able to balance exploration and exploitation. In this work we will focus on the UCB criterion, as in our problem we prefer to have the ability to tune the balance of exploration and exploitation.

C. Bayesian Optimization with Gaussian Process Regression

Bayesian optimization consists of two main components: a Bayesian statistical model that represents the objective, and an acquisition function for deciding where to sample next. After evaluating the objective according to an initial space-filling experimental design, often consisting of points chosen uniformly at random, they are used iteratively to allocate the remainder of a budget of N function evaluations.

Figure 3 shows an example of using Bayesian optimization on a 1D optimization problem [23]. The figures show a Gaussian process (GP) approximation of the objective function over four iterations of sampled values of the objective function. The figure also shows the acquisition function as green shaded plots. The acquisition value is large where the

GP predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration)—areas with both attributes are sampled first. Note that the area on the far left remains unsampled, as while it has high uncertainty, it is (correctly) predicted to offer little improvement over the highest observation.

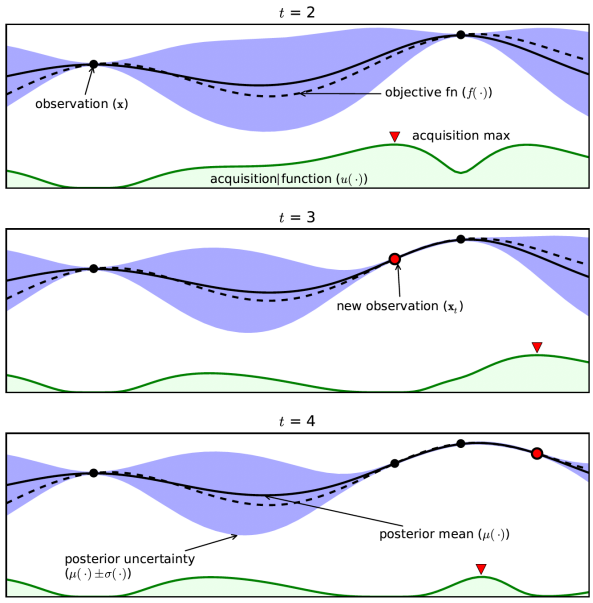


Fig. 3 An example of using Bayesian optimization on a toy 1D optimization problem [23].

IV. Problem Formulation

The aim of this work is to address the following questions:

1) Given a UTM decision making system, can we automatically discover what scenarios lead to that system performing poorly?
2) Given a set of specified use cases, can we optimize the variables parameterizing our decision making system to improve its performance?

To formulate this problem, consider a UTM system which consists of two parts: the scenario parameterized by the variables x_s which represents the planned flight requests, the decision making system parameterized by the variables y_s which is responsible for generating path waypoints for the whole trajectory of the planned flight requests. Given x_s and y_s we can run simulations to evaluate a configuration of our decision making system on a given scenario. We represent the metric we care about by m. Formally, we have:

$$m = f(\mathbf{x_s}, \mathbf{y_s}) \tag{5}$$

where f is the function we'd like to approximate so we can determine the value of interest m given x_s and y_s.

In this work we answer question one by performing a simulation of scenarios parameterized by x_s under decision making system parameterized by y_s. Our goal is to find a subset(s) of x_s that lead to some range of values m given a fixed trajectory planning algorithm with parameters y_s. These will likely result in scenarios that are unsafe or inefficient, i.e. through simulation we determine they fall below some safety and/or efficiency threshold. To answer question two, our goal is to find y_s that optimizes m for a given subset of x_s. In other words, we want to find the hyper-parameters that lead to the best performing algorithm for a given set of scenarios. The above two tasks can be formulated into optimization problems and in this work we will use the Bayesian Optimization algorithm to solve the formulated optimization problem. The advantage of Bayesian Optimization is sample efficiency (the algorithm will search the most promising variable configuration given the previous information), since the function f is expensive to evaluate as we must perform a simulation to do so.

A. Failure Mode Discovery

Our first task is as follows: given a parameterized autonomous system in the UTM ecosystem, we try to identify its failure modes by maximizing the objective function that correlates with system failure. For the strategic deconfliction services considered in this work, we construct the objective function using the sum of all NMACs in a scenario weighted by their severity:

$$f = \sum_{i=1}^{N_{\text{nmac}}} s_i \tag{6}$$

where s_i, the severity of NMAC i, is a function of its duration and the proximity of the other vehicle in conflict. The optimization problem then takes the following form:

$$\begin{aligned} \max \quad & f(\mathbf{x_s}; \mathbf{y_s}) && \text{(7a)} \\ \text{subject to} \quad & g_i(\mathbf{x_s}) \leq 0, \;\; i = 1, 2, \ldots, n_{ineq} && \text{(7b)} \\ & h_i(\mathbf{x_s}) = 0, \;\; i = 1, 2, \ldots, n_{eq} && \text{(7c)} \end{aligned}$$

where g_i and h_i are generic functions that capture any inequality and equality constraints the scenario parameterization x_s is subject to. In practice, we apply constraints to remove infeasible scenarios from the optimization search space, and convert equality constraints into inequality constraints when performing Bayesian Optimization. Note that in the optimization above, we are optimizing only over the scenario space $\mathbf{x_s}$, and not over the parameters of the system under test $\mathbf{y_s}$. This optimization discovers failures scenarios $\mathbf{x_s}$ for a given instantiating of the system parameterized by $\mathbf{y_s}$.

B. Exploration and Exploitation Trade-offs

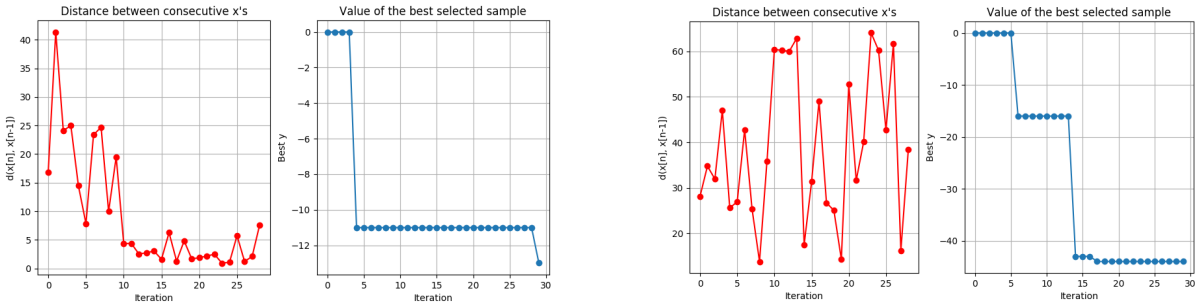
The exploration and exploitation trade-off is a critical part of the methodology, and we address it in this section. By using the Upper Confidence Bound (UCB) [22] as the acquisition function in the Gaussian process optimization, we can tune the parameter responsible for this trade-off (κ in Equation 4). Fig. 4 shows the effects of the exploration constant during an example stress test run for a UTM service responsible for strategic deconfliction. We explore the details behind this example further later in the paper. The left plot in red represents the Euclidean distance between consecutive sampled points in the scenario space. A small distance means that the point sampled next is close to the previously sampled point. Smaller inter-point distances tend to result in low variance of the samples. A large distance on the other hand implies a robust exploration of the parameter space. As shown in Fig. 4 (a), the algorithm gets stuck in a local optimum from sample 4 to sample 28 while performing minimization, where the discovered failure scenarios have a numeric value of negative eleven. While in Fig. 4 (b), the algorithm quickly finds new failure scenarios (at sample 6, 14, 17) instead of getting stuck in a local optimum. As we compare Fig. 4 (a) and (b), it can be seen with a higher exploration weight (e.g., $\kappa = 10$), the optimization algorithm tends to focus on areas where the uncertainty is high. This is a useful technique for identifying new failure modes through Bayesian optimization as opposed to staying stuck in a local optimum which corresponds to a single failure mode. To ensure an exploration driven optimization, we set the exploration constant $\kappa = 10$.

C. System Optimization

Next we consider the problem of optimizing a system for a given set of scenarios that lead to system failures. This optimization problem has the form:

$$\begin{aligned} \min \quad & f(\mathbf{y_s} \mid \mathbf{x_s}) && \text{(8a)} \\ \text{subject to} \quad & l_i(\mathbf{y_s}) \leq 0, \;\; i = 1, 2, \ldots, n_{ineq} && \text{(8b)} \\ & m_i(\mathbf{y_s}) = 0, \;\; i = 1, 2, \ldots, n_{eq} && \text{(8c)} \end{aligned}$$

where l_i and m_i capture the infeasibility constraints associated with the parameters of the system we are optimizing. While the objective is still computed using Equation 6, it is now being minimized in an effort to reduce or eliminate the failure modes of the system. Note that x_s takes on the form of a random variable in the optimization above that is sampled from a distribution of scenarios D_f that led to system failures $x_s \sim D_f$.



(a) $\kappa = 2$ **(b)** $\kappa = 10$

Fig. 4 An example of the step distances and objective values in Bayesian optimization with different exploration weights.

Table 1 Operation parameters for the UAM and package delivery use cases.

Use Case	Cruise Speed (m/s)	Cruise Altitude (m)	Separation Requirement (m)
Delivery	15	115	100
UAM	60	500	150

By applying the two methodologies above in tandem, we can perform failure mode discovery and system optimization to those failure modes in a single min-max optimization. This process leads to a streamlined system and scenario level analysis of complex and difficult to reach parts of the scenario and system parameter spaces.

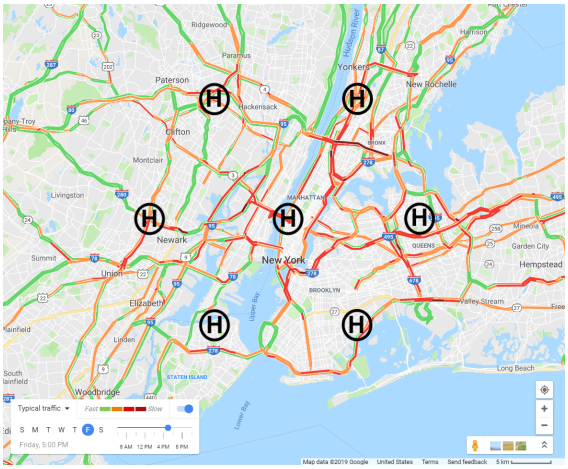
D. Simulated Use Cases

In this work, we evaluate our approach using simulated UAM and package delivery use cases. The primary differences between each use case are their operational profiles and the dynamic envelopes of the vehicles used to operate them. For the UAM use case we assume an operational profile that traverses within a network of fixed vertiports, while for the package delivery use case we assume operations that traverse between a centralized fulfillment center and a randomly generated delivery destination. The geographic configuration of each use case are shown in Fig. 5a. The UAM use case follows a generic city model presented in [24–26] with seven vertiports distributed in a "six around one" hexagonal pattern. The distance of the central vertiport from the other six vertiports is 16 km. Overlays of the vertiport network are shown on a Google map image of New York City, which shows typical New York traffic on a Friday at 5pm [27]. The geographic configuration of the package delivery use case is shown in Fig. 5b. There are two fulfillment centers in the region separated by 12 km, and overlaid over a map of Toulouse.

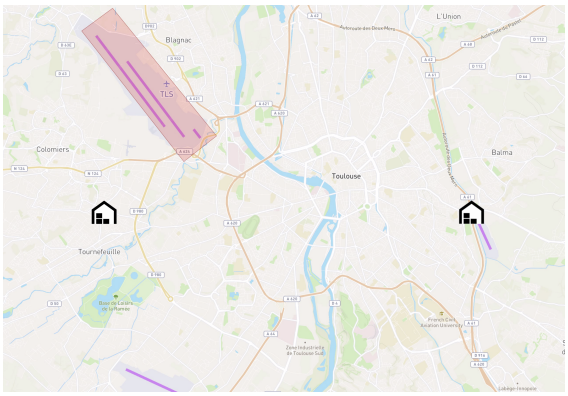
We simulate each use case on a scenario by scenario basis, with each scenario parameterized by the UTM operations within it. We focus on the ability of the UTM services being stress tested to perform strategic or pre-flight deconfliction. We assume in-flight disturbances are minimal, and thus operations that are properly deconflicted from one another pre-flight will adhere to the separation standards enforced through pre-flight deconfliction in flight. Each operation can be described by a unique set of discrete and continuous variables that represent the operation start time, its origin, and its destination, which we refer to as scenario variables $\mathbf{x_s}$ in Eq. 8 and Eq. 7. For the UAM use case the origin and destination variables are represented by seven discrete values of the vertiports in the UAM network. For the delivery use case, the origin is represented by the two discrete values of the fulfillment centers, while the destination is represented by a lat-long pair that falls within the service area of the fulfillment center. Additionally, each operation has a set of properties that are unique to the use case which include cruise speed, cruise altitude, and required separation as shown in Table 1. These properties are not part of the scenario parameterization in this work, so they are held constant across use cases during stress testing.

We assume a first-come, first-served (FCFS) resource allocation between operations in the simulated scenarios. This implies that operations that were submitted into the system first, get priority. Practically, this leads to later submitted operations needing to deconflict with operations submitted before them to ensure separation. While implications of the FCFS approach have been recently examined in the context of UTM [28, 29], they fall outside the scope of the adaptive

stress testing methodology evaluated in this work. We leave the connection between UTM resource allocation policies and system failures to future work, and instead focus on system failures that result from logic errors within individual UTM services.



(a) Urban Air Mobility



(b) Package Delivery

Fig. 5 Simulated use cases evaluated in adaptive stress testing.

V. Numerical Experiments

In this section we present the numerical results of applying the adaptive stress testing algorithm to two distinct strategic deconfliction services. While the services have identical objectives of ensuring strategic separation between operations, each is driven by different algorithms. The first uses 4D trajectory optimization, and deconflicts operations spatially from one another, without optimizing departure time. We refer to it as the trajectory deconfliction service. The second performs dynamic scheduling by solving a linear program, and deconflicts operations through departure delay. We refer to it as the scheduling service. We apply the adaptive stress testing methodology to these services while they are used to deconflict simulated operations for the Urban Air Mobility and the package delivery use cases. For both sets of stress tests, we use a single valued objective function from Equation 6. All scenarios sampled in the evaluation contain two operations, and we leave many operation strategic deconfliction validation for future work. We conclude this section by presenting the sample efficiency gains of our approach over a naive methodology that attempts to discover system failures using uniform sampling.

A. Trajectory Deconfliction Stress Testing

This subsection presents the results that apply the stress testing approach outlined in this paper to a trajectory deconfliction UTM service for the UAM and package delivery use cases. We also present results that show how the deconfliction service can be improved with respect to the failure modes by optimizing the hyper-parameters governing the algorithm that is responsible for the logic within the service. The system we are testing is a variant of the trajectory optimization algorithm proposed in [30], which optimizes an operational 4D trajectory with respect to a numerical cost objective and a set of dynamic and geo-spatial constraints. The algorithm takes flight request(s) as input and returns a single or a set of optimized 4D trajectories that separate the vehicles according to the prescribed separation standard. The algorithm itself does not provide theoretical guarantees that the trajectories generated will be conflict free. With the help of the adaptive stress testing algorithm we are able to discover precisely which scenarios lead to conflicting operations and subsequent system failures.

We apply Gaussian process regression as outlined in Section IV.A to the two use cases of interest. We define a single stress test as an optimization that samples 150 points from the respective scenario spaces of each use case. The run-time optimization metrics for the UAM use case are shown in Fig. 6. The optimization is divided into two phases, an exploratory phase that samples scenarios uniformly for the first 50 samples, and an optimization phase that maximizes the acquisition function in Equation 4 for the next 100 samples. During the exploratory phase, the distance

Table 2 Failure mode types of the trajectory deconfliction service, capability of the system optimization step, and potential mitigations of the failure mode if the system optimization is not able to resolve the failure.

Failure Type	Discovered In Use Case	Resolved Through Optimization	Failure Mitigations
During Take-Off	Both	No	Procedures/Scheduling
During Landing	Both	No	Procedures/Scheduling
Flying Directly Towards	UAM	Yes	-
Flying in Sequence	UAM	Yes	-
Crossing Traffic	Both	Yes	-

between consecutively sampled points is large since the algorithm is picking uniformly from the scenario space. While distances are still large between points for samples 50 through 60, we see a dramatic drop off in the distance following the discovery of the first failure mode around sample 60. At that point the algorithm has discovered a failure mode in the system and begins to exploit the scenario space around the region of discovery. A number of additional failure modes are discovered soon after the initial failure discovery. We run an identical stress test using the package delivery scenario, and perform a classification of the failure modes discovered in each optimization process. We found that trajectory deconfliction failures tend to occur in one of the following five types: during take-off, during landing, flying directly towards one another, flying in sequence, crossing traffic. The relevant use cases for each type of failure mode are shown in Table 2. Note that the UAM use case has two failure modes that were not found in the package delivery use case.

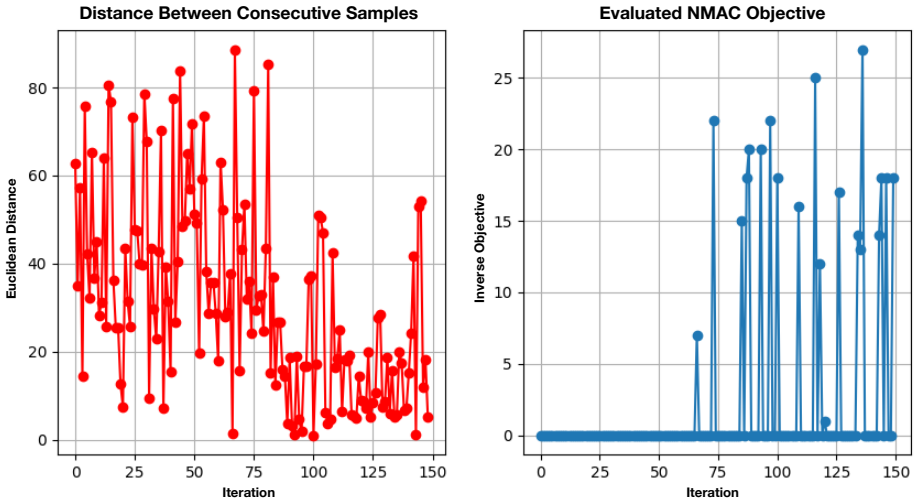


Fig. 6 Run-time optimization metrics for the UAM use case

Next, we perform a parameter optimization of the algorithm behind the trajectory deconfliction service in an effort to mitigate the failures through a re-parameterization of the system. The optimization is done with respect to the discovered failure modes, and its intention is to optimize the algorithm hyper-parameters in a way that resolves the discovered system failures. Formally, we perform the sequential optimizations that solve Eq. 7 and Eq. 8. The variable hyper-parameters in the algorithm take on the following form:

$$\mathbf{y_s} = [l_{\max}, l_{\min}, C_{constraint}] \tag{9}$$

where $l_{\max}, l_{\min}$ are the maximum/minimum bounds of a cruise flight segment in the generated trajectory. Whereas $C_{constraint}$ is the penalty function multiplier associated with creating optimization constraints with respect to other existing operations from which deconfliction is required. Readers can refer to [30] for more details about the algorithm and the hyper-parameters associated with it.

The $\mathbf{y_s}$ defined in Eq. 9 can be optimized by solving Eq. 8. In this work we used Bayesian optimization to also solve

Eq. 8. A qualitative outline of the results for the two step algorithm that performs stress testing followed by failure mode optimization is shown in Table 2. Specifically, the results for whether the system optimization was successful in resolving the failure mode or not are shown in addition to a proposed failure mitigation mechanism when the hyper-parameter optimization was unable to resolve the system failure. For the two classes of failures where the system optimization was not successful, we propose additional mitigation mechanisms, and demonstrate their validity in the following section. Because the two failures that remained unresolved occur during take-off or landing, we propose that these two conflict types be resolved using procedural separation or a scheduling services.

The system level optimization for the trajectory deconfliction service hyper-parameters led to the following changes in the parameter settings: l_{max} was left unchanged at 500 m, l_{min} was modified from 50 m to 30 m, $C_{constraint}$ was modified from 50 to 500. Two examples of how the changes due to optimization above that successfully resolved the flying in sequence and crossing traffic failure modes are shown in Fig. 7. The top row of the figure shows how a scenario that requires two operations to fly in sequence led to a failure due to sub-optimal size of trajectory segments created by the trajectory deconfliction service. The bottom row shows how a scenario that requires two operations to cross lead to a failure due to a low penalty cost for constraints imposed by existing operations during trajectory optimization in the deconfliction service.

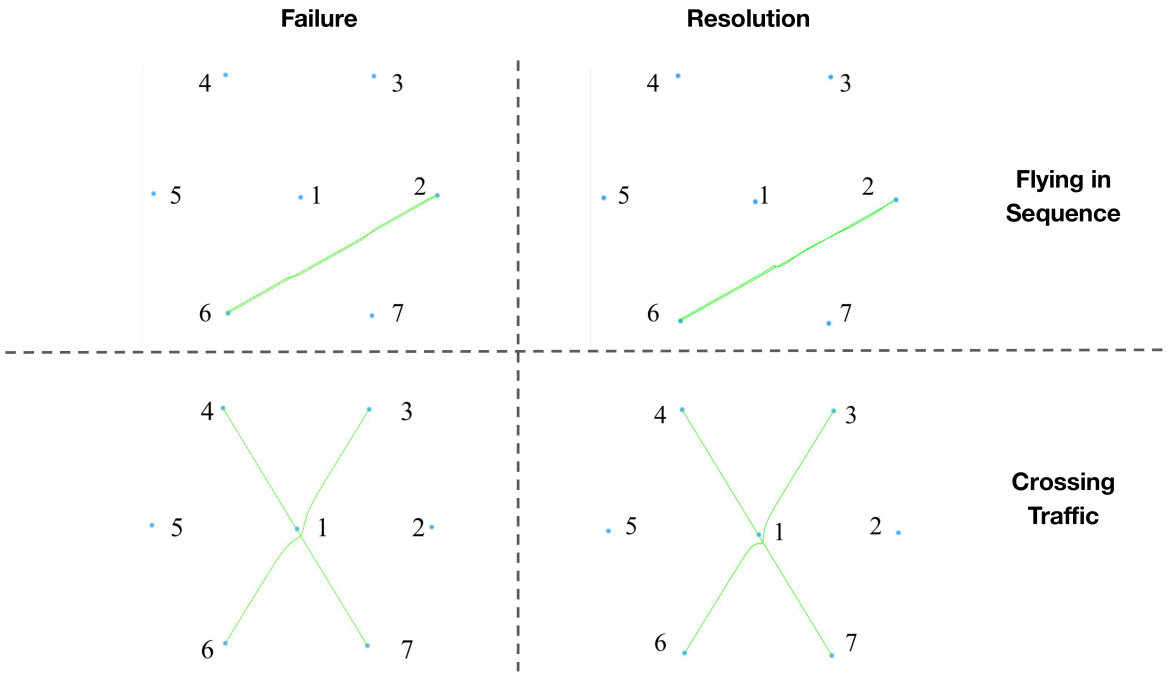


Fig. 7 Examples of system parameter optimization that successfully resolved two strategic deconfliction failure modes.

B. Scheduling Service Stress Testing

This subsection presents the results that apply the stress testing approach to the scheduling service for UTM operations. The scheduling service logic is driven by an algorithm that solves a linear program which attempts to minimize the delay of a single operation in the presence of existing operations that create constraints in the time domain. As such, the service takes in an operation proposal in the form of a 4D trajectory as input and returns a potentially delayed 4D trajectory that is conflict free from operations that already exist in the UTM system. The linear program solver relies on additional components to perform deconfliction. Namely, a component that initializes the 4D trajectory from an operation request, a component that transforms the proposed operation and existing operations into a set of linear constraints, and a component that transforms the output of the linear program into a 4D trajectory with potential delays.

Upon applying the adaptive stress testing methodology to the scheduling service, we discovered an infrequently

occurring failure mode in both the UAM and the package delivery use cases. This occurred during take-offs for the delivery use case and during take-offs and landings for the UAM use case. Further investigation uncovered an error in the trajectory initialization component which led to silent failures in the linear program meeting all the constraints (see Figure 8). While these conditions were rare and only occurred when the start time of a proposed operation coincided exactly with the end time of the first flight segment of a conflicting operation, they ultimately resulted in NMACs during the simulation. Despite the primary goal of the adaptive stress testing methodology being validation of the decision logic in the strategic deconfliction service, we consider the discovery of an error in the supporting component of the service a successful application of its capability. The methodology played an important role in identifying a critical error in the system being validated that would have been difficult for a subject matter expert to identify on their own.

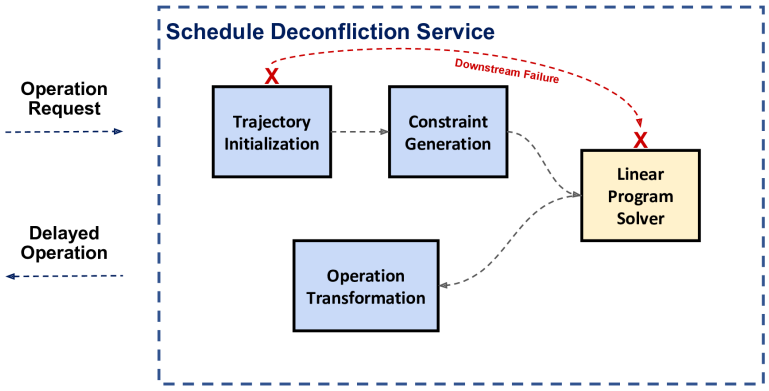


Fig. 8 Downstream failure propagation in the scheduling service.

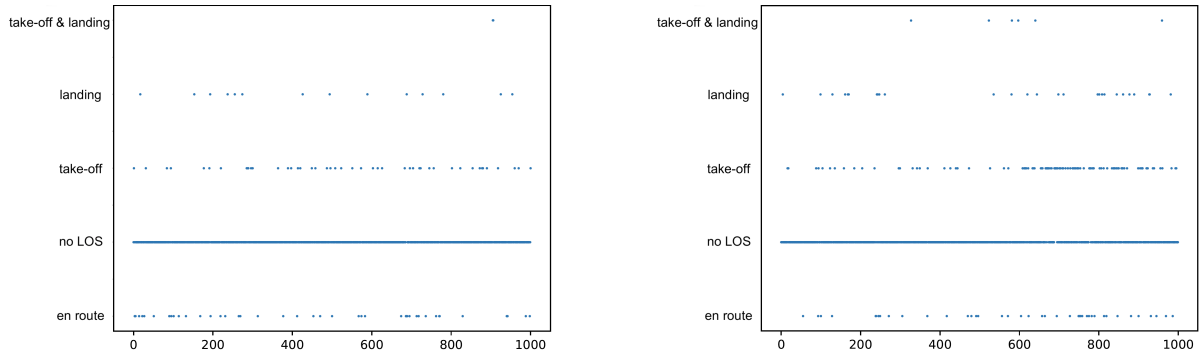
C. Sampling Efficiency

To further demonstrate the benefits of the adaptive stress testing approach, we evaluate its sampling efficiency. We compare our algorithm with a baseline algorithm that uniformly samples validation scenarios from the UAM use case scenario space and tests the trajectory deconfliction service before it has undergone hyper-parameter optimization. We are thus comparing the ability of each approach to discover system failures. In this experiment, the Bayesian optimization algorithm and baseline algorithm both sample 1,000 scenarios, with the results shown in Figure 10. The figure shows the run-time categories of the scenarios sampled by each algorithm. Notice that while many of the sampled scenarios do not result in a system failure, and have no loss of separation (LOS) events, a number of sampled scenarios do. While it is not clear from the plot the quantitative differences between the scenario samples of the two approaches, it is important to note the significant increase in failure mode samples generated by the Bayesian optimization adaptive stress tester. This is likely due to the algorithm exploiting a learned property of the failure mode scenarios it has sampled previously while still performing significant exploration of the scenario space.

A quantitative comparison of the two algorithms is shown in Figure 10 and Table 3. The figure shows a fractional breakdown of the sampled scenarios by category for the two algorithms in question. The table, on the other hand, illustrates the improvement ratio in samples that the Bayesian adaptive stress testing approach had over the uniform sampling approach for each category. We see a significant increase in the number of sampled scenarios that result in system failures when the Bayesian adaptive stress testing approach is used. A nearly 2× improvement in failure samples are observed for system failures that occur at take-offs, landings, and take-off & landings. The number of samples for en route failures is nearly identical between the two approaches.

VI. Conclusion

In this work, we proposed a novel adaptive stress testing algorithm based on Bayesian optimization, and demonstrated how it can be applied to validate advanced UTM services that enable autonomous strategic deconfliction. We showed that the algorithm can be used in simulated use cases that cover package delivery and UAM. Additionally, we demonstrated the ability of the methodology to discover scenarios that lead to system failures, which in this context meant scenarios



(a) Uniform sampling. **(b) Bayesian optimization.**

Fig. 9 Failure modes identified by different sampling strategies as a function of scenario sample number.

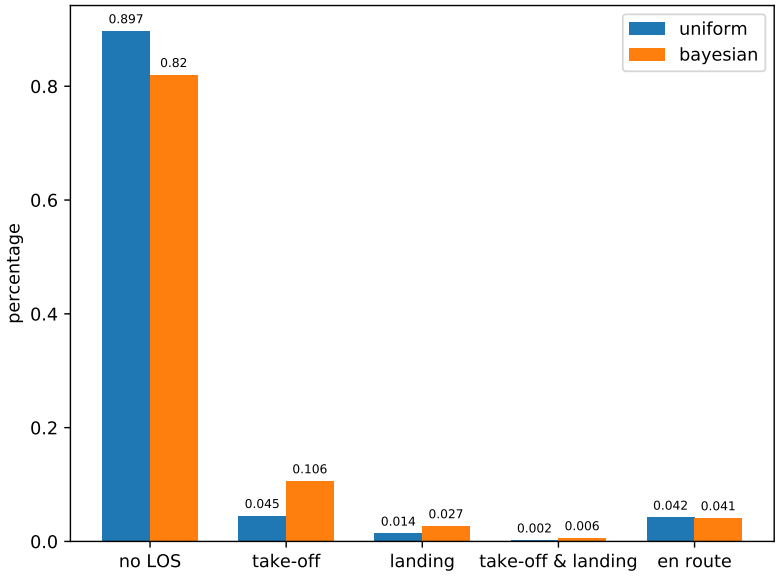


Fig. 10 Failure modes fractions by category for the uniform sampling and the Bayesian optimization adaptive stress testing approaches.

that ultimately resulted in NMACs and a failure in strategic deconfliction. Ultimately, we demonstrated the value of the adaptive stress testing tool, and how it can be integrated into the validation process of a UTM service responsible for decision making in the airspace. We provided examples of its benefits over validation methodologies that do not use simulation based validation, and over naive simulation based validation approaches

Future work beyond this paper includes scaling the Bayesian adaptive stress testing methodology to scenarios with many operations, and to incorporate in-flight disturbances and uncertainties into the validation evaluation. Additionally, future work will consider more complex UTM services and environments such as services that combine scheduling and trajectory deconfliction as well as using these algorithms in services that manage in-flight or tactical separation. The methodology will also be extended to consider and validate interoperability between service providers by modeling multiple operators and decision making services that may have different algorithmic logic. Interoperability validation is of particular importance in the context of a federated UTM which may require automated negotiation, which has implications on resource allocation and fair airspace usage [28, 29]. Lastly, the theoretical properties of our approach will be examined, which will be critical for understanding how the exploration in the Bayesian adaptive stress testing approach relates to other black box safety validation algorithms [31], and to approaches that have been successfully applied to accelerate validation of safety-critical systems in aviation such as importance sampling [13].

Table 3 The ratio of increased sample counts for each scenario category between the uniform and the Bayesian optimization adaptive stress testing validation strategies.

	No LOS	Take-off	Landing	Take-off & Landing	En Route	Total Failures
Improvement Ratio	0.91	2.36	1.93	3.0	0.98	1.75

References

[1] Balakrishnan, K., Polastre, J., Mooberry, J., Golding, R., and Sachs, P., "Blueprint for the Sky: The Roadmap for the Safe Integration of Autonomous Aircraft," *Airbus UTM, San Francisco, CA*, 2018.

[2] Jenkins, D., Vasigh, B., Oster, C., and Larsen, T., *Forecast of the commercial UAS package delivery market*, Embry-Riddle Aeronautical University, 2017.

[3] Holden, J., and Goel, N., "Uber Elevate: Fast-Forwarding to a Future of On-Demand Urban Air Transportation," *Uber Technologies, Inc., San Francisco, CA*, 2016.

[4] Hamilton, B. A., "Urban Air Mobility Market Study," , 2018. URL `https://go.nasa.gov/2MVSbth`.

[5] Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., and Robinson, J. E., "Unmanned aircraft system traffic management (UTM) concept of operations," 2016.

[6] ICAO, "ICAO Document 9683: Human Factors Training Manual," , 2006. [Online].

[7] Ball, T., and Rajamani, S. K., "Automatically validating temporal safety properties of interfaces," *Proceedings of the 8th international SPIN workshop on Model checking of software*, Springer-Verlag, 2001, pp. 103–122.

[8] Gardner, R. W., Genin, D., McDowell, R., Rouff, C., Saksena, A., and Schmidt, A., "Probabilistic model checking of the next-generation airborne collision avoidance system," *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–10.

[9] Katoen, J.-P., "The probabilistic model checking landscape," *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, ACM, 2016, pp. 31–45.

[10] Platzer, A., *Logical analysis of hybrid systems: proving theorems for complex dynamics*, Springer Science & Business Media, 2010.

[11] Ghetiu, T., Polac, F. A., and Bown, J., "Argument-driven validation of computer simulations-a necessity, rather than an option," *2010 Second International Conference on Advances in System Testing and Validation Lifecycle*, IEEE, 2010, pp. 1–4.

[12] Morton, J., Wheeler, T. A., and Kochenderfer, M. J., "Closed-loop policies for operational tests of safety-critical systems," *IEEE Transactions on Intelligent Vehicles*, Vol. 3, No. 3, 2018, pp. 317–328.

[13] Kochenderfer, M. J., Edwards, M. W., Espindle, L. P., Kuchar, J. K., and Griffith, J. D., "Airspace encounter models for estimating collision risk," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 487–499.

[14] Lee, R., Kochenderfer, M. J., Mengshoel, O. J., Brat, G. P., and Owen, M. P., "Adaptive stress testing of airborne collision avoidance systems," *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, IEEE, 2015, pp. 6C2–1.

[15] Koren, M., Alsaif, S., Lee, R., and Kochenderfer, M. J., "Adaptive stress testing for autonomous vehicles," *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1–7.

[16] Lee, J., Shin, H., and Shim, D. H., "Path-planning with collision avoidance for operating multiple Unmanned Aerial Vehicles in same airspace," *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 1460.

[17] Mockus, J., *Bayesian approach to global optimization: theory and applications*, Vol. 37, Springer Science & Business Media, 2012.

[18] Jones, D. R., "A taxonomy of global optimization methods based on response surfaces," *Journal of global optimization*, Vol. 21, No. 4, 2001, pp. 345–383.

[19] Snoek, J., Larochelle, H., and Adams, R. P., "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[20] Rasmussen, C. E., "Gaussian processes in machine learning," *Summer School on Machine Learning*, Springer, 2003, pp. 63–71.

[21] Kushner, H. J., "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, Vol. 86, No. 1, 1964, pp. 97–106.

[22] Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M., "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[23] Brochu, E., Cora, V. M., and De Freitas, N., "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[24] Kohlman, L. W., and Patterson, M. D., "System-level urban air mobility transportation modeling and determination of energy-related constraints," *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3677.

[25] Patterson, M. D., Antcliff, K. R., and Kohlman, L. W., "A Proposed Approach to Studying Urban Air Mobility Missions Including an Initial Exploration of Mission Requirements," 2018.

[26] Yang, X., and Wei, P., "Scalable Multi-Agent Computational Guidance with Separation Assurance for Autonomous Urban Air Mobility," *Journal of Guidance, Control, and Dynamics*, 2020. doi:10.2514/1.G005000.

[27] Google, "Google Maps," , 2019. URL `https://www.google.com/maps`, accessed: 2019-03-12.

[28] Evans, A. D., Egorov, M., and Munn, S., "Fairness in Decentralized Strategic Deconfliction in UTM," *AIAA Scitech 2020 Forum*, 2020, p. 2203.

[29] Chin, C., Gopalakrishnan, K., Evans, A., Egorov, M., and Balakrishnan, H., "Tradeoffs between Efficiency and Fairness in Unmanned Aircraft Systems Traffic Management," *9th International Conference on Research in Air Transportation*, 2020.

[30] Egorov, M., Kuroda, V., and Sachs, P., "Encounter Aware Flight Planning in the Unmanned Airspace," *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, IEEE, 2019, pp. 1–15.

[31] Corso, A., Moss, R. J., Koren, M., Lee, R., and Kochenderfer, M. J., "A Survey of Algorithms for Black-Box Safety Validation," *arXiv preprint arXiv:2005.02979*, 2020.