OpenMDAO: Framework for Flexible Multidisciplinary Design, Analysis and Optimization Methods

Christopher M. Heath[1] and Justin S. Gray[2]
NASA Glenn Research Center, Cleveland, OH, 44135

The OpenMDAO project is underway at NASA to develop a framework which simplifies the implementation of state-of-the-art tools and methods for multidisciplinary design, analysis and optimization. Foremost, OpenMDAO has been designed to handle variable problem formulations, encourage reconfigurability, and promote model reuse. This work demonstrates the concept of iteration hierarchies in OpenMDAO to achieve a flexible environment for supporting advanced optimization methods which include adaptive sampling and surrogate modeling techniques. In this effort, two efficient global optimization methods were applied to solve a constrained, single-objective and constrained, multi-objective version of a joint aircraft/engine sizing problem. The aircraft model, NASA's next-generation advanced single-aisle civil transport, is being studied as part of the Subsonic Fixed Wing project to help meet simultaneous program goals for reduced fuel burn, emissions, and noise. This analysis serves as a realistic test problem to demonstrate the flexibility and reconfigurability offered by OpenMDAO.

ASAT	=	advanced single aisle transport
BLISS	=	bi-level integrated system synthesis
CM-EGO	=	constrained multi-objective efficient global optimization
CO	=	collaborative optimization
CS-EGO	=	constrained single-objective efficient global optimization
D_L	=	landing field length
D_{TO}	=	take-off field length
EGO	=	efficient global optimization
EI	=	expected improvement
$F_{N,mapp}$	=	missed approach excess net thrust
$F_{N,SLS}$	=	sea level static net thrust
$F_{N,SS}$	=	second segment climb net thrust
FPR_{ADP}	=	fan pressure ratio at aerodynamic design point
GA	=	genetic algorithm
$\dot{h}_{pot,\ toc}$	=	potential rate of climb at top of climb conditions
IDF	=	individual design feasible
LHC	=	Latin hypercube
LTO	=	landing take-off cycle
MDAO	=	multidisciplinary design, analysis and optimization
MDF	=	multidiscipline design feasible
NASA	=	National Aeronautics and Space Administration
NM_{cum}	=	cumulative noise margin relative to the stage 4/chapter 4 rule
NSGA-II	=	non-dominated sorting genetic algorithm (version II)
NO_x	=	oxides of nitrogen
PI	=	probability of improvement
S_W	=	reference trapezoidal wing area
UHB	=	ultrahigh bypass
V_{app}	=	approach velocity
$W_{blockfuel}$	=	block fuel weight
$W_{excessfuel}$	=	excess fuel weight
W_{GTO}	=	gross takeoff weight

[1] Aerospace Engineer, Multidisciplinary Design, Analysis and Optimization Branch, MS 5-10, Non-member.
[2] Aerospace Engineer, Multidisciplinary Design, Analysis and Optimization Branch, MS 5-10, Non-member.

I. Introduction

One of the recognized challenges in the multidisciplinary design, analysis and optimization (MDAO) field today is the need for reconfigurability and reusability of analysis techniques[1]. The OpenMDAO project is a NASA initiative to develop an open-source framework for meeting this need, among others. OpenMDAO seeks to provide users with a set of common tools and interface that aids in the setup and solution of complex engineering design, analysis and optimization problems. Special emphasis is placed on framework development to ensure robustness (capacity to support the most advanced of research optimization methods) and flexibility (capacity to test different optimization approaches across many problem formulations and multiple computing platforms).

OpenMDAO is being written in the Python programming language and leverages the object-oriented format to decompose MDAO algorithms. This modular design ensures flexibility, allowing different optimization methods to be applied to many problem formulations. New optimization schemes can be added to the framework by defining new classes and the performance of various optimizers can be readily compared. To date, OpenMDAO provides support for several complex optimization architectures including collaborative optimization (CO), multidisciplinary design feasible (MDF), individual design feasible (IDF), bi-level integrated system synthesis (BLISS), and other variants. The standard OpenMDAO library currently provides access to several gradient and non-gradient optimizers, as well as options for performing design of experiments (DOEs) and applying various surrogate modeling techniques. This paper focuses on the implementation of efficient global optimization (EGO)[2-5], an adaptive sampling architecture, and applies EGO to a realistic NASA aircraft/engine sizing problem.

II. Adaptive Sampling

There currently exist a large number of MDAO methods which make use of adaptive sampling, surrogate models and Bayesian statistics to locate areas of the design space which appear valuable, as measured by an objective function. These algorithms employ an iterative scheme where each true function call improves the quality of surrogate models used to approximate the design space. Rather than optimize the true problem directly, the optimizer searches the approximate design space for regions where closer refinement or investigation is statistically likely to result in the discovery of optimal points. These methods are often said to balance exploration and exploitation, as they carefully explore the entire design space, exploiting regions where optimal designs are likely to occur. The general premise is to minimize the total number of expensive function calls by relying on surrogate model approximations during the optimization process. This makes adaptive sampling techniques most ideal for problems involving computationally expensive function evaluations. Adaptive sampling has been researched since the mid 90's[6-9], with some of the earliest work involving Bayesian methods done in the late 70's by Mockus[10]. Despite ongoing research in the area, these techniques have not been widely adopted by today's MDAO frameworks. For this reason, adaptive sampling has been infrequently applied to solve real world MDAO problems in engineering. Two instances of frameworks that do provide support for adaptive sampling are DAKOTA[11] and, more recently, OpenMDAO. OpenMDAO enables both single and multi-objective adaptive sampling, while DAKOTA is known to support only a single-objective version at this time.

To enable single and multi-objective adaptive sampling, OpenMDAO introduces the "iteration hierarchy" and "MetaModel" software concepts. Iteration hierarchies allow complex processes, like those often found in MDAO architectures, to be easily defined. Rather than require a new class for each optimization method, an iteration hierarchy can be created and modified at runtime. This greatly enhances the flexibility of the problem setup, allowing users to modify MDAO approaches on demand. Similarly, to simplify the use of surrogate modeling in optimization algorithms, OpenMDAO utilizes a special analysis component called "MetaModel". A meta-model is a software tool used to automatically mimic inputs and outputs for any given software component. This means that a meta-model can seamlessly replace an expensive software code with an inexpensive approximation. Matching of component inputs and outputs by a meta-model greatly streamlines the process of implementing surrogate modeling to solve MDAO problems.

In the remaining sections of this paper, we document the fundamental design elements of OpenMDAO, explaining how the concepts of "iteration hierarchy" and "MetaModel" have been integrated into the framework. We demonstrate how a meta-model component can be used within an iteration hierarchy to iteratively train surrogate models. Those surrogate models in turn are used to implement the EGO algorithm. We show how OpenMDAO permits a high degree of reusability between single and multi-objective versions of the EGO algorithm, and how the problem can be reconfigured easily for either formulation. As a proof-of-concept, OpenMDAO implementations of the single-objective and multi-objective EGO algorithm are presented. Both EGO versions are exercised on a representative NASA aircraft/engine sizing problem.

III. ASAT Problem Formulation

The NASA advanced single-aisle transport (ASAT) model is an example of a real-world multidisciplinary design problem as it combines analyses of airframe aerodynamics, turbine engine thermodynamics, weight estimation, acoustic signature, and mission performance. Similar system-level aircraft optimizations have been performed previously by Antoine and Kroo[12,13]. The ASAT model was first presented by Guynn[14] where several single-objective optimizations were performed to map the feasible design space. The analysis was later continued by Berton and Guynn[15], where a multi-objective optimization was performed using the non-dominated, sorting genetic algorithm (NSGA-II)[16].

The ASAT airframe is based on the current 162-passenger Boeing 737-800, with several next-generation technology upgrades assumed and commensurate with a 2015 service entry date. The design Mach number for this vehicle is 0.8 and the design range is 3,250 nm, assuming a payload of 32,000 lb. Another technology enhancement is the moderate use of composite structures on up to 50% of the airframe, resulting in a 15% reduction in component weight for the wing, fuselage and empennage[15]. Also, an increase in hydraulic pressure to 5000 psi and 1% drag reduction, due to variable trailing edge camber and drag clean-up were assumed.[15] In the original problem formulation, the airframe was fitted with a pair of ultrahigh bypass (UHB) turbofan engines, producing a number of discrete propulsion-related design variables. Previous research indicates that high bypass engines show promise in reducing noise and emissions while simultaneously improving overall propulsive efficiency.[17] Engine options included direct drive versus gear driven fan, fixed versus variable area bypass nozzle, high verses low overall pressure ratio, and high verses low work split between the low pressure and high pressure spools. For simplicity, we pre-selected an engine architecture subset from Guynn and Berton's work that included a direct drive fan, high overall pressure ratio, high work split and a fixed area bypass nozzle. This eliminated complexity associated with extending EGO to handle discrete variables, but substantially reduced the size of the design space. Efforts have been made to construct EGO formulations that permit discrete design variables, but this research is fairly recent and still ongoing.[18] Figure 1 displays the planform view of NASA's notional ASAT model. Additional details on the next-generation vehicle are provided in Ref. 15.

Figure 1. Planform view of NASA's notional advanced single aisle transport concept.

In Guynn and Berton's research, the NSGA-II genetic optimization ran for 198 complete generations and analyzed a total 9,504 discrete designs[15]. Each individual aircraft/engine analysis took approximately 2 minutes to run on a single workstation. This yielded a total execution time for the optimization of approximately 13 days.

For the ASAT problem, EGO seems to be a less costly optimization alternative. The EGO process begins with a sparse initial sampling of the design space, followed by an iterative adaptive sampling based on expected improvement (EI) or probability of improvement (PI). During the adaptive sampling, Pareto optimal points from the current set of sampled designs are extracted and used to calculate EI and PI for the entire objective space. A global optimizer searches this space for the point expected to most greatly improve the Pareto frontier. For this research, optimization based on PI and EI was tested. PI-based optimization was found to perform better, giving slightly improved results. For this reason, we present results obtained only using PI to adaptively sample the design space. In this implementation, EGO makes use of kriging surrogate models to minimize the number of design evaluations necessary to find the Pareto frontier. An added benefit of EGO is that it does not rely on the use of an aggregate objective (i.e. weighted sum or utility function). This enables the technique to find Pareto optimal solutions regardless of whether or not the Pareto front is convex.

We have applied constrained single-objective EGO (CS-EGO) and constrained multi-objective EGO (CM-EGO) to the NASA ASAT optimization problem. Both CS-EGO and CM-EGO are implemented using iteration hierarchies in OpenMDAO. The formal optimization statement as provided in Ref. 15 is:

Minimize objectives:

$$f_1: \frac{W_{block\ fuel}(lb)}{30000},\quad f_2: \frac{NM_{cum}(EPNdB)}{25},\quad f_3: \frac{W_{GTO}(lb)}{150000},\quad f_4: \frac{LTO_{NOx}(g/kN)}{20}$$

With respect to parameters:

$$x_1: \frac{S_W(ft^2)}{1400},\quad x_2: \frac{F_{N,SLS}(lb)}{26000},\quad x_3: FPR_{ADP}$$

Within the ranges of:

$$0.8 \leq x_1 \leq 1.6,\quad 0.8 \leq x_2 \leq 1.6,\quad 1.35 \leq x_3 \leq 1.7$$

Subject to constraints:

$$g_1(x): 1 - \frac{D_{TO}(ft)}{7000} \geq 0$$

$$g_2(x): 1 - \frac{D_L(ft)}{7000} \geq 0$$

$$g_3(x): 1 - \frac{V_{app}(knots)}{150} \geq 0$$

$$g_4(x): \frac{\dot{h}_{pot,toc}(ft/min)}{300} - 1 \geq 0$$

$$g_5(x): \frac{W_{excess\ fuel}(lb)}{10000} \geq 0$$

$$g_6(x): \frac{F_{N,SS}(lb)}{1000} \geq 0$$

$$g_7(x): \frac{F_{N,mapp}(lb)}{1000} \geq 0$$

All design parameters and objectives have been made dimensionless and are scaled by an appropriate constant to be of equivalent orders of magnitude. Objectives f_1 through f_4 correspond to the block fuel burned ($W_{block\ fuel}$), cumulative stage 4 noise margin (NM_{cum}), gross take-off weight (W_{GTO}) and landing take-off cycle NO_x emissions ($LTO\ NO_x$), respectively. Design variables x_1 and x_2 represent the size of the wing (S_W) and sea-level static engine thrust ($F_{N,SLS}$), while x_3 is a propulsion system design parameter for the fan pressure ratio at the aerodynamic design point (FPR_{ADP}). Inequality constraints g_1 through g_7 are typical aircraft/engine sizing requirements for take-off field length (D_{TO}), landing field length (D_L), approach velocity (V_{app}), potential climb rate at top-of-climb ($\dot{h}_{pot,toc}$), excess fuel weight ($W_{excess\ fuel}$), excess thrust for second-segment climb ($F_{N,SS}$) and excess thrust for missed approach ($F_{N,mapp}$).[15]

Berton and Guynn examined a number of sub-problems from the above formulation. They separately examined objectives f_1 and f_3 as single-objective problems and separated the four objectives into three, two-objective problems: f_1 with f_2, f_2 with f_3, and f_3 with f_4. For this research, we also analyzed a single-objective sub-problem, minimizing ramp weight (objective f_3). To be consistent with the prior multi-objective analysis results, we similarly decomposed the four objective problems into three two-objective problems.

IV. Algorithm Implementation in OpenMDAO

This section details the approach used to implement EGO in OpenMDAO. We first clarify the iteration hierarchy concept along with surrogate modeling. We then demonstrate the use of data objects to emphasize reconfigurability and MDAO algorithm reuse.

A. Iteration Hierarchy

OpenMDAO is built around four basic programming class constructs: *Component*, *Assembly*, *Driver* and *Workflow*. A Component class instance accepts input variables and returns output after performing a calculation. An assembly instance, which also inherits from the Component base class, is a container object which houses one or more child components, defining input and output data connections between them. Data connections specify the direction data may flow between components. By default, all assemblies contain at least one driver. The Driver class is used to control process iteration. All driver instances contain a single workflow. A workflow is a container object for components, assemblies and other drivers that dictates which child components a driver should execute and order of execution. Drivers iterate over their workflow until a predefined stopping condition is met. Since the Driver class also inherits from the Component class, the nesting of one driver inside the workflow of another creates an iteration hierarchy[19]. Note that an iteration hierarchy describes process flow independent of data flow. No information is contained regarding the exchange of data between software components. We represent data dependencies separately using a data flow diagram. The iteration hierarchy enables many complex research optimization algorithms to be represented and implemented in OpenMDAO. The CS-EGO and CM-EGO algorithms are two such examples that we present in the following sections. Although the specifics of the problem formulations differ, both share the common iteration hierarchy seen in Fig. 2. Note the use of the "ASAT MetaModel" component to approximate ASAT model outputs from a given set of model inputs. Other details of this iteration hierarchy as they apply to using EGO to solve the ASAT problem will be discussed in the upcoming sections.

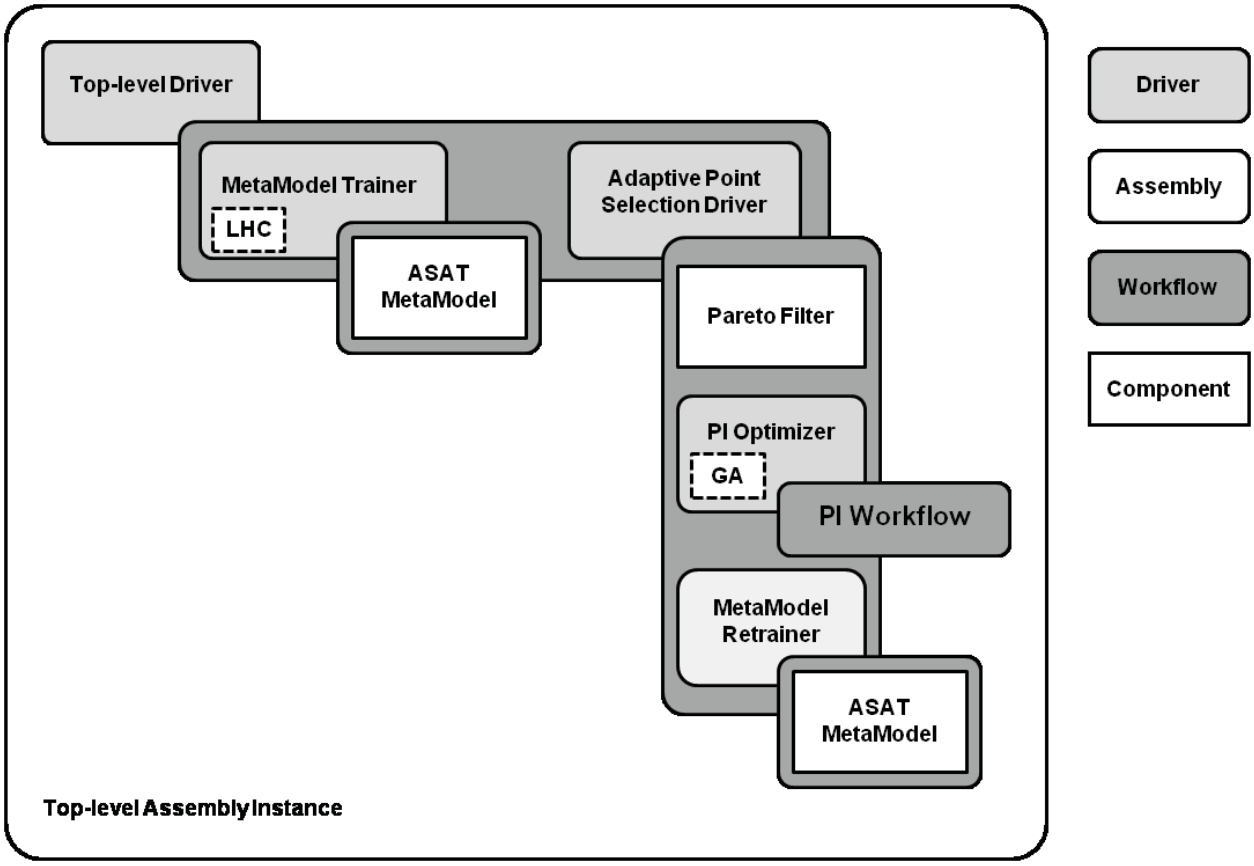

Figure 2. Iteration hierarchy in OpenMDAO for the ASAT aircraft/engine sizing problem.

B. Surrogate Modeling

In OpenMDAO, the term "surrogate model" refers to a mathematical approximation which represents the behavior of a single output of a component. Since most components have more than one output, we use the term "meta-model" to refer to the collection of surrogate models corresponding to all outputs of a given component.

The OpenMDAO standard library provides the *MetaModel* component for generic meta-model generation. MetaModel contains two slots, "model" and "surrogate", which together provide the flexibility to create meta-models of any component using any surrogate modeler. In OpenMDAO, a slot is simply a placeholder for a component plugin which serves to provide a common interface. MetaModel has no inherent inputs or outputs of its own. Instead, MetaModel mimics all inputs and outputs of the slotted model. This behavior allows a meta-model to seamlessly replace any component instance. Figure 3 shows a meta-model instance, called "ASAT MetaModel", with the model slot filled by the NASA ASAT model. The inputs and outputs of "ASAT MetaModel" look identical to those of the NASA ASAT model, including model outputs for objective functions and constraints. A surrogate modeler interface has also been specified in OpenMDAO. Any object implementing the surrogate modeler interface can be plugged into the surrogate slot. This slot specifies the kind of surrogate models MetaModel will create. In Fig. 3, "Kriging Surrogate" fills this slot, so that MetaModel will create kriging approximations of all outputs from the NASA ASAT model.

For the ASAT aircraft/engine sizing problem, a kriging meta-model is used to approximate the design space for the adaptive sampling process. To do this, a set of aircraft/engine designs determined by a space-filling DOE are analyzed. All meta-models in OpenMDAO have the ability to be trained by a DOE. By activating a meta-model training event, inputs passed to a meta-model instance execute the underlying model contained in the model slot. Outputs from the execution of each design in the DOE are stored to a database file. After all cases from the DOE complete, an event reads the database results back in and generates the underlying surrogates needed to create the meta-model instance. The data set from the DOE forms the training data set for the NASA ASAT meta-model component. Once training data has been collected, a meta-model component instance can be used in place of an actual analysis model to predict outputs for a given set of model inputs. To get predicted values, valid inputs must be supplied to the meta-model and the meta-model component must be executed in non-training mode. Inputs are passed to the underlying surrogate models for each output and the predicted responses are pushed to the meta-model outputs. We will later show how the semantics of this design allow for flexibility when applying the EGO algorithm to a design problem.

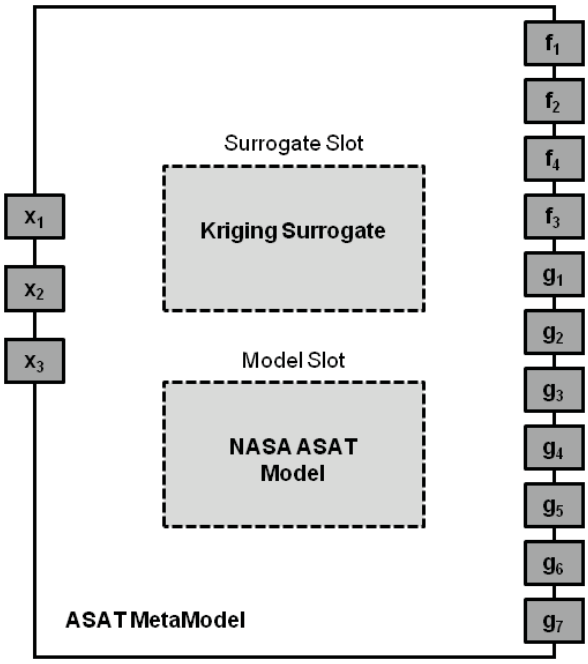

Figure 3. ASAT MetaModel instance with kriging surrogate model and NASA ASAT model in the respective slots. Inputs and outputs from the ASAT model are mimicked at the MetaModel boundary.

C. Efficient Global Optimization in OpenMDAO

For this subsection, we refer the reader back to Figs. 2 and 3. The EGO algorithm, as implemented in OpenMDAO, begins with a top-level assembly instance as shown in Fig. 2. This assembly contains a default driver, named "Top-level Driver", which manages the top-level workflow of the assembly. Two sub-drivers named "MetaModel Trainer" and "Adaptive Point Selection Driver" are shown in the top-level driver workflow. "MetaModel Trainer" is actually a DOE driver which houses the "ASAT MetaModel" component in its workflow and executes first. Similar to MetaModel, DOE drivers in OpenMDAO contain a slot for which any DOE generator may be connected. In this case, an optimized Latin hypercube (LHC) DOE generator occupies the slot, and the DOE is executed on the "ASAT MetaModel" component, with the training event option activated. Input and output from DOE points are stored to a database file and used to train the Bayesian models of the kriging surrogate. Once training of the kriging model completes, the workflow of the "Adaptive Point Selection Driver" is executed. "Adaptive Point Selection Driver" houses three sub-components in its workflow: a "Pareto Filter" component, a "PI Optimization" driver, and a "MetaModel Retrainer" driver. "Adaptive Point Selection Driver" controls the number of optimization iterations of its workflow and terminates when either the maximum achievable PI falls below a

predefined limit or the maximum number of iterations for the workflow are exceeded, whichever comes first. Before each iteration of "Adaptive Point Selection Driver", the current set of data used for training the ASAT MetaModel is sent through "Pareto Filter", where the cases are sorted according to the problem objective (or objectives). Information regarding the location of non-dominated designs is necessary to calculate PI prior to the optimization.

For the single-objective EGO problem, the "Pareto Filter" component sorts the training cases in decreasing order according to objective value. A single case is found for what Forrester refers to as the "best observed point so far"[20]. This point is passed to the "Single-Objective PI Calculator" component shown in the PI workflow of Fig. 4. Note, this sub-workflow is actually the PI workflow of the "PI Optimizer" driver in Fig. 2. The "Single Objective PI Calculator" and "PI_{g1}" through "PI_{g7}" are all component instances of the *ExpectedImprovement* class in OpenMDAO. The "Single Objective PI Calculator" instance returns PI for the objective space based on the kriging model approximation, while "PI_{g1}" through "PI_{g7}" return PI for each constraint.

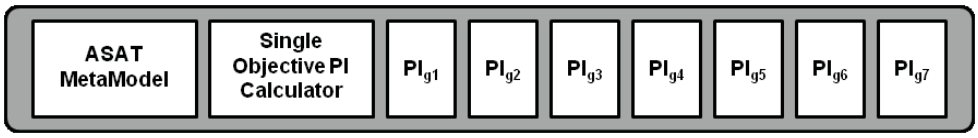

Figure 4. PI workflow for the single-objective problem formulation.

For the multi-objective formulation, all objectives are added to the "Pareto Filter" component. Due to the multiple objectives, the Pareto filter finds the complete set of Pareto optimal points from the current training data set. These points serve as the "best observed points so far" for the "Multi-Objective PI Calculator" component in Fig. 5. To perform multi-objective EGO, this workflow simply replaces the single-objective workflow in Fig. 4. Similarly, this workflow also plugs-in to the common EGO iteration hierarchy shown in Fig. 2. Note the single and multi-objective PI workflows differ only in the use of either a single or multi-objective PI calculator component. The "Multi Objective PI Calculator" is a component instance of the *MultiObjExpectedImprovement* class in OpenMDAO. This instance also returns the PI for the objective space. Components "PI_{g1}" through "PI_{g7}" remain instances of the ExpectedImprovement class and return PI for each constraint.

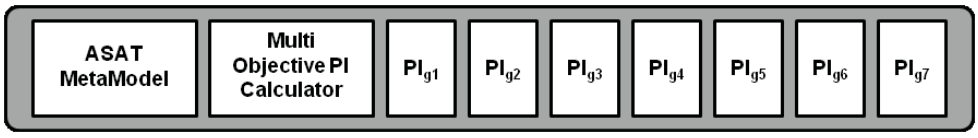

Figure 5. PI workflow for the multi-objective problem formulation.

After Pareto cases from the current data set have been obtained by the "Pareto Filter" component, the "PI Optimization" driver is executed using either the single or multi-objective "PI Workflow" in Fig. 4 or 5. For this problem, the *Genetic* optimizer from the OpenMDAO standard library was used as the "PI Optimization" driver, but any global optimization driver could easily be substituted. The Genetic Algorithm (GA) maintained a population size of 200 and was set to terminate after 50 generations. Crossover was based on tournament selection and design variables were not binary encoded. The PI workflow contains another instance of the "ASAT MetaModel" component. This instance differs from the previous one used in the "MetaModel Trainer" workflow, in that the MetaModel training event is deactivated. This means that all calls to this "ASAT MetaModel" instance will result in approximate outputs as determined by the underlying surrogates of "ASAT MetaModel". The goal of the global optimizer is to inexpensively search the PI space (using surrogate approximations) for the set of design variables which maximize PI of the objective(s), subject to the given problem constraints.

The mathematical form of the PI optimization objective is given in Eq. 1. $PI(X)$ represents the calculated value for probability of improvement at location X in the design space. $PI_{gn}(X)$ is the probability of improvement of the constraint value for constraint n at the location X in the design space.

$$f_{PI} = PI(X) * PI_{g1}(X) * PI_{g2}(X) * PI_{g3}(X) * PI_{g4}(X) * PI_{g5}(X) * PI_{g6}(X) * PI_{g7}(X) \quad (1)$$

After the PI optimization completes, the location in the design space with the best overall constrained PI is identified. The "MetaModel Retrainer" driver then executes the new "ASAT MetaModel" training case, with the training event option again reactivated. This triggers an actual function call to the ASAT model, adding new meta-

model training data to the set of aircraft/engine designs previously run. The underlying surrogate models of the ASAT MetaModel component are updated and the PI optimization process is repeated starting back at the "Pareto Filter" component of the "Adaptive Point Selection Driver" workflow. The "Adaptive Point Selection Driver" continues to run, adding new meta-model training points every iteration, until either the maximum achievable PI falls below 1×10^{-10} or a maximum of 150 iterations is exceeded by the driver.

The iteration hierarchies for the single-objective and multi-objective algorithms are largely identical. This commonality is only made possible through the use of the MetaModel class in OpenMDAO, which handles the job of surrogate model generation for each output of the NASA ASAT model. If this were not the case, significant portions of workflows would need to change to accommodate the training of different surrogate models, one for each output included in the multi-objective problem formulation.

Up until this point, we have discussed the process flow of EGO, neglecting the flow of data between individual software components. Each "ASAT MetaModel" instance appearing in the workflow of Fig. 2 contains the data connections shown in Fig. 6. OpenMDAO manages the flow of data and information is exchanged between components only when necessary. A set of single-objective PI component instances, one for each constraint (g_1 through g_7), are common between both single and multi-objective EGO problem formulations. In this case, both PI components (multi-objective and single-objective instances) are dependent on output from the "ASAT MetaModel" component. The "Single-Objective PI Calculator" component is connected to only the f_3 output from "ASAT MetaModel" because we have elected to use only ramp weight, or f_3, for the single objective analysis. Objectives f_1 through f_4 are passed as an input array to the "Multi Objective PI Calculator" where they are aggregated and used to calculate the multi-objective PI. Since execution is controlled by the workflow in OpenMDAO, data connections can be established between the objective outputs from "ASAT MetaModel" and both PI instances, without interference.

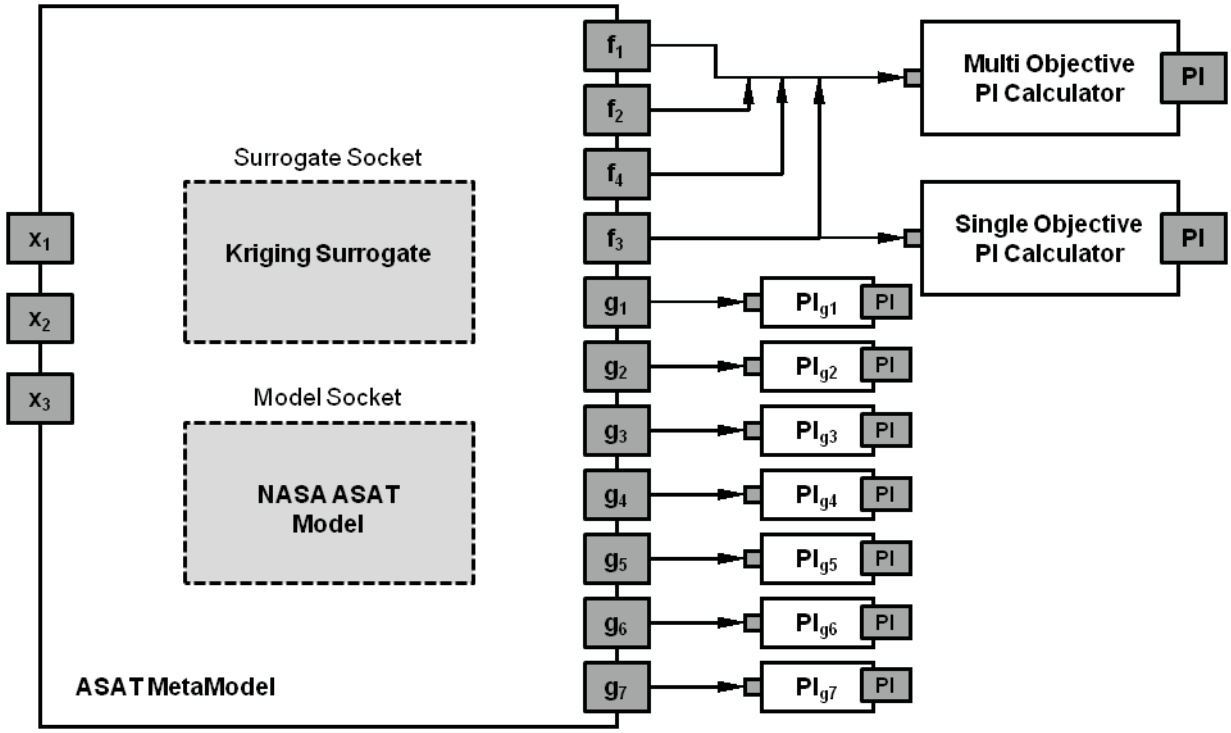

Figure 6. Data connection diagram for CS-EGO and CM-EGO algorithms.

Specifying the single-objective PI workflow in Fig. 4 or the multi-objective PI workflow in Fig. 5 tells the "PI Optimizer" driver which problem formulation (single or multi-objective) to execute. Within OpenMDAO, the flexibility exists to configure data connections to components which may or may not be used in an MDAO algorithm. This permits generation of a robust MDAO model that can support multiple problem formulations, single and multi-objective, simultaneously.

It is important to note that the "ASAT MetaModel" component shows up three times in the iteration hierarchy. "ASAT MetaModel" appears once in the workflow of "MetaModel Trainer", once in the "PI Workflow" and once in the workflow of "MetaModel Retrainer". This represents the exact same component appearing at three different

locations in the MDAO process, and each time being executed by a different driver. This offers the flexibility to have one driver train the surrogate models of "ASAT MetaModel", while a second is used for optimization on the predicted values from "ASAT MetaModel", and a third performs additional surrogate model training as new points are adaptively sampled by the optimizer.

V. Constrained Single-Objective Optimization Results

Results based on the CS-EGO formulation for W_{GTO} are presented in Fig. 7. In the scatter plot, designs denoted by gray symbols were identified during the adaptive sampling process. Designs marked with black symbols indicate initial training data from the LHC DOE. The data show that the majority of cases were run by the DOE, with fewer than 30% of the designs adaptively sampled by the optimizer. Infeasible designs, or cases in which one or more constraints were violated, are plotted as triangles. Designs corresponding to failed cases have been excluded. For this particular problem, a failed case is one in which an aircraft/engine combination fails to produce a set of valid outputs. Errors have the potential to occur inside any of the individual engineering codes that make up the ASAT model as a result of numerical instabilities or simply poor designs. Convergence failures were common, such that regions of the design space were found to be infeasible and frequently produced failed cases.

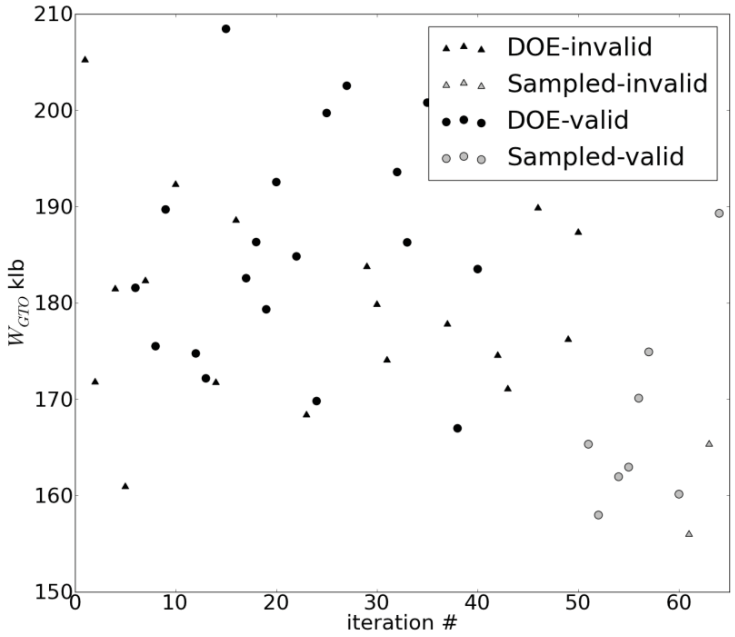

Figure 7. Single-objective EGO results. Gray points are from the initial sample DOE. Black points are adaptively sampled points. Triangles are infeasible designs.

For ASAT model code failures, Berton chose to report artificial, but extremely high, objective function values. This helped eliminate failed designs from the population of the genetic algorithm. A similar approach was tried for EGO, but led to several problems. Foremost, adding artificially high objective values to the kriging training data set caused the surrogate model quality to degrade, making the overall optimization ineffective to the point of not outperforming a random search. To avoid this, we tested a scheme of simply omitting failed cases from the current training set. This approach was intended to rely on the non-deterministic nature of the GA to slightly vary the location of the global optimum on successive iterations. Unfortunately, this method was also unsuccessful. When the optimization ventured into an infeasible region of the design space, small perturbations were not enough to achieve a converged ASAT model solution and the optimizer failed to advance. Forrester et al. proposed a compromise solution, where failed cases were replaced with the predicted outputs from the kriging surrogate for that output, adjusted to account for the relative error at that point[21]. Due to the modular construction of the optimization approach in OpenMDAO, implementing this method required only one minor change to the code involving the training of the kriging surrogate. The need to try a number of different methods for dealing with failed cases emphasized the benefits of a modular architecture for defining optimization algorithms.

Of the 50 DOE cases used for initially training the "ASAT MetaModel", 40% (20 cases) were feasible solutions, 38% (19 cases) were infeasible solutions, while the remaining portion were failed cases. As evident in Fig. 7, the minimum W_{GTO} identified was 158,125 lbs. Following the 50 case initial sample set, 14 adaptively sampled points (8 feasible, 2 infeasible, 4 failed) were selected and executed by the optimizer. The optimization terminated when the PI value dropped below the predefined threshold of 1×10^{-10}. The minimum weight solution found was 4.8% greater than the minimum weight design of 150,800 lbs found by Berton[15]. In terms of an aircraft design problem, this weight difference is quite significant. However, the computational cost of performing the analysis using EGO was considerably less. The data in Fig. 7 also show that the minimum W_{GTO} was found early in the sampling process and a higher threshold for minimum PI may have been appropriate. The low values for PI during sampling also

indicate that the space may have been over sampled by the DOE, making EGO less effective. When a large number of initial sample points are used, the uncertainty of the kriging surrogate models becomes reduced, decreasing the sensitivity of the PI calculations.

Any of the other objectives, f_1, f_2, or f_4, could have been similarly optimized instead by simply changing a single data connection between the "ASAT MetaModel" and the "Single-Objective PI Calculator" component.

VI. Constrained Multi-Objective Optimization Results

The formulation of the ASAT problem with objectives f_1, f_2, f_3 and f_4 produced the results shown in Figs. 8, 9 and 10. Fig. 8 plots objective NM_{cum} versus $W_{blockfuel}$ (f_2 vs. f_1), Fig. 9 shows objective NM_{cum} versus W_{GTO} (f_2 vs. f_3), and Fig. 10 plots *LTO* NO_x versus W_{GTO} (f_4 vs. f_3). For consistency, the same 50 case LHC DOE was applied to solve all three multi-objective problem formulations. In each case, the DOE seemed more appropriately sized for initializing the kriging model. In the scatter plots, designs denoted by gray symbols were identified during the adaptive sampling process. White symbols represent Pareto optimal points, or designs in which one objective cannot be improved without compromise to the other. Designs marked with black symbols indicate initial training data from the LHC DOE. Infeasible designs, or cases in which one or more constraints were violated, are plotted as triangles. Again, designs corresponding to failed cases have been excluded from all plots. Overall, the data serve to highlight important trade-offs between objectives and qualitatively compare well with results from Ref. 15. Note for the single engine architecture analyzed in this study, Pareto optimal points tend to be dominant for all objectives. This is an interesting result, as it was previously believed that advancements in NO_x could not be achieved without degradation or severe compromise to at least one other objective.

Quantitatively comparing these findings to previous results found with the NSGA-II optimizer is somewhat more challenging. The overall trends are consistent, but the Pareto frontier was much more densely sampled by Berton and Guynn. In addition, the manner in which Berton and Guynn presented data for various engine architectures further makes direct comparison difficult. We refer the reader to Refs. 14 and 15 for a more detailed description of the relationships between objectives for the ASAT optimization problem.

In general, the optimizer heavily sampled local regions of the objective space that were close to being Pareto optimal. This is evident by the large clustering of points in only a few locations of the objective space. The optimizer attempted to exploit regions of the design space where it appeared significant improvements could be made. For most objectives, few cases investigated designs that were far from the Pareto frontier. Further refinement of the scaling terms for each objective may have helped improve the overall performance of CM-EGO on this problem.

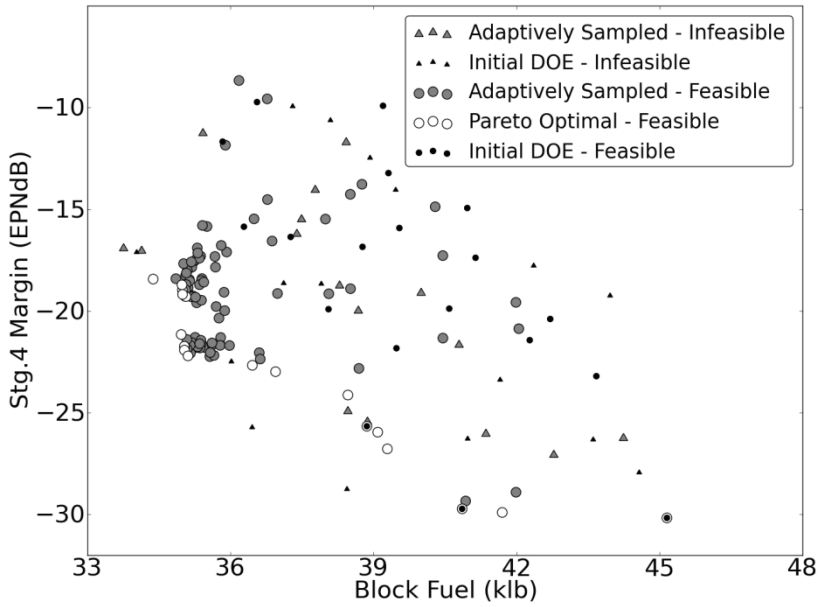

Figure 8. Multi-objective EGO results for cumulative noise margin relative to the stage 4/chapter 4 rule versus block fuel weight.

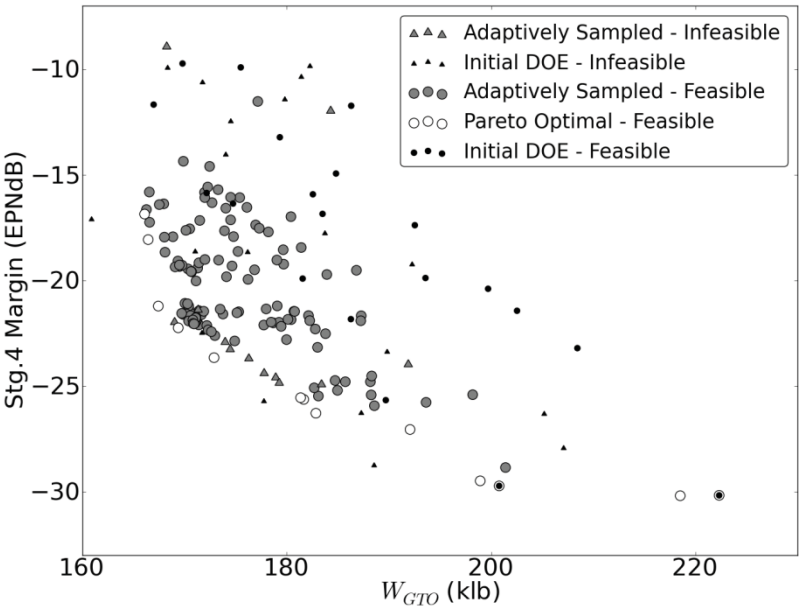

Figure 9. Multi-objective EGO results for cumulative noise margin relative to the stage 4/chapter 4 rule versus gross take-off weight.

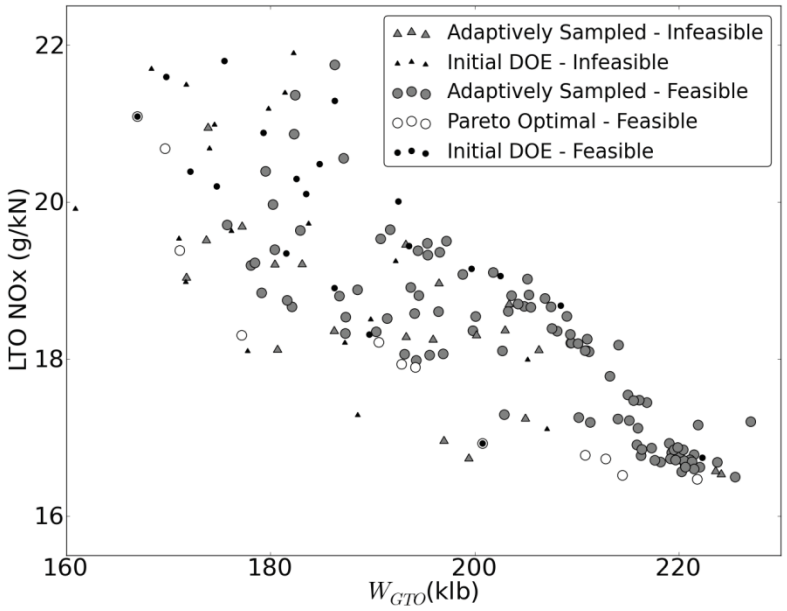

Figure 10. Multi-objective EGO results for the landing take-off cycle NOx versus the gross take-off weight.

VII. Discussion

Overall, data obtained through this study compare well with previously reported findings, validating the performance of the EGO algorithms on this particular MDAO problem. Berton and Guynn performed a more comprehensive study to analyze additional discrete engine architecture effects including variable nozzle exit geometries and the use of geared turbofans that were not considered here. Since their design space was significantly larger, a quantitative performance comparison is not possible. Regardless, some basic qualitative comparisons can be made.

The initial sampling DOE seemed to provide the algorithm with sufficient training data to make effective kriging surrogate models of all objectives and constraints. As a result, over 80% of the adaptively sampled points that executed successfully were feasible solutions. In Berton and Guynn's results, approximately 30% of the sampled designs were feasible.[15] This shows a dramatic improvement in the number of feasible designs investigated by the optimizer. In general, the EGO algorithm was successful at optimizing the ASAT model for both the single-objective and multi-objective problem formulations given the relatively small number of actual function calls. For the CS-EGO formulation, fewer than 70 actual function evaluations were needed to find a reasonable optimum design. For CM-EGO, two hundred cases were run for each of the three problem formulations. Since the same 50 sample points (20 of which were feasible solutions) from the DOE were re-used to initially train the kriging model in each CM-EGO analysis, only 500 cases total were run between all three CM-EGO formulations. The low number of cases alone does not accurately represent the total optimization computational cost. The kriging surrogate model training and the PI-based optimization both consumed a significant portion of the optimization time. When function calls take hours or days, the overhead of the kriging models and PI optimization become less significant. For this problem, the time for a single function evaluation and PI algorithm overhead were roughly equal. Total optimization time took approximately twice as long as the number of cases would suggest, assuming each case took on average two minutes to run. Even so, with a computational cost equivalent to about 1000 actual function evaluations, CM-EGO proved to be an effective algorithm for the ASAT concept study.

VIII. Conclusions

EGO served as a powerful example of how a complex MDAO algorithm can be decomposed and implemented in OpenMDAO using iteration hierarchies. In addition, the problem setup in OpenMDAO was easily reconfigured to accommodate single objective and multi-objective formulations of the ASAT analysis. EGO is comprised of a relatively simple set of data connections, but with a detailed workflow which accomplishes the iterative process: Pareto filtering, training point selection, and the addition of the new training data to the meta-model. We identified the major commonality between the CS-EGO and CM-EGO algorithms and preserved that commonality in the problem setup within OpenMDAO. Other than reconfiguring the objectives, the two configurations differ only by the workflow of the "PI Optimization" driver.

We successfully applied CS-EGO and CM-EGO algorithms to the NASA ASAT problem, which combines analyses from multiple disciplines to identify a set of conceptual aircraft solutions that minimize fuel burn, emissions and noise. The results show that reconfigurability built into an MDAO framework provides greater flexibility in selecting a MDAO approach to efficiently solve problems. That approach can later be extended to encompass more design metrics, in the form of multi-objective optimization, without significant changes to the MDAO process.

IX. Future Work

Development of OpenMDAO is ongoing, and the latest release versions of the software can be found on the project website at http://openmdao.org. Future development efforts will include implementing other MDAO tools like new surrogate modeling techniques, optimization methods and DOEs. Research is ongoing to extend the CM-EGO algorithm in OpenMDAO to include discrete variable types. This work is focusing on the application of EGO to system-of-systems design problems involving early selection for conceptual design.

X. Acknowledgements

The authors gratefully acknowledge the financial support of the Subsonic Fixed Wing Project in NASA's Aeronautics Research Mission Directorate.

References

[1]Alexandrov, Natalia. M., and Lewis, Robert M., "Reconfigurability in MDO Problem Synthesis, Part 1,"August 2004, Paper AIAA 2004-4307.

[2]Jones, D. R., Schonlau, M., and Welch, W. J., "Efficient Global Optimization of Expensive Black-Box Functions". Journal of Global Optimization, Vol. 13, 1998, pp. 455–492.

[3]Forrester, A. I., and Keane, A. J., "Recent Advances in Surrogate-Based Optimization". Progress in Aerospace Sciences, Vol. 45, Issues 1-3, 2009, pp. 50-79.

[4]Basudhar, A., Lacaze, S., and Missoum, S., "Constrained Efficent Global Optimziation with Probabilisitc Support Vector Machines". 13th AIAA/ISSMO Multidisciplinary Analysis and Optimzation Conference, 2010, AIAA 2010-9230.

[5]Forrester, A. I., and Jones, D. R., "Global Optimization of Deceptive Functions with Sparse Sampling". 2nd AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008, Paper AIAA 2008-5996.

[6]Watson, A. G., Barnes, R. J., "Infill Sampling Criteria to Local Extremes," Mathematical Geology, Vol. 27, Issue 5, 1995, pp. 589–608.

[7]Cox, Dennis. D., and John, Susan.,"SDO: A Statistical Method for Global Optimization". Multidisciplinary Design Optimization: State-of-the-Art. Institute for Computer Applications in Science and Engineering, 1997, pp. 315–329.

[8]Hawe, G. I., and Sykulski, J. K., "A Scalarizing One-Stage Algorithm for Efficient Multi-objective Optimization".IEEE Transactions on Magnetics, Vol. 44, Issue 6, 2008, pp. 1094–1097.

[9]Villemonteix, J., Vazquez, E., andWalter, E., "An Informational Approach to the Global Optimization of Expensive to Evaluate Functions". Journal of Global Optimization, Vol. 44, Issue 4, 2009, pp. 509–534.

[10]Mockus, J., Tiesis, V., and Zilinskas, A., "The Application of Bayesian Methods for Seeking the Extremum". Towards Global Optimization, Vol. 2, 1978, pp. 117–129.

[11]Bichon, B. J., Eldred, M. S., Swiler, L. P., Mahadevan, S., and McFarland, J. M., "Multimodal Reliability Assessment for Complex Engineering Applications Using Efficient Global Optimization". 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2007, AIAA 2007-1946.

[12]Antoine, N. E., and Kroo, I. M. "Aircraft Optimization for Minimal Environmental Impact". Journal of Aircraft, Vol. 41, Issue 4, 2004, pp. 790-797.

[13]Antoine, N. E., and Kroo, I. M. "Framework for Aircraft Conceptual Design and Environmental Performance Studies". AIAA Journal, Vol. 43, Issue 10, October 2005, pp. 2100-2109.

[14]Guynn, M. D., Berton, J. J., Fisher, K. L., Haller, W. J., Tong, M. T., and Thurman, D. R., "Analysis of Turbofan Design Options for an Advanced Single-Aisle Transport Aircraft". 9th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2009, Paper AIAA 2009-6942.

[15]Berton, J. J., and Guynn, M. D.,"Multi-objective Optimization of Turbofan Design Parameters for an Advanced, Single-Aisle Transport". In 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010, Paper AIAA 2010-9168.

[16]Deb, K., Pratap, A., Agarwal, S., and T., M., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II". IEEE Transactions on Evolutionary Computation, Vol. 6, 2002, pp. 182–197.

[17]Envia, Ed., "Propulsion Noise Reduction Concepts and Progress". Green Aviation Summit, No. E-17590, NASA Ames Research Center, September 2010.

[18]Rousis, Damon A., "A Pareto Frontier Intersection-Based Approach for Multiobjective Optimization of Competing Concept Alternatives". Ph.D. Dissertation, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 2011.

[19]Gray, J., and Moore, K. T., "Openmdao: An Open-source Framework for Multidisciplinary Analysis and Optimization". 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2010, Paper AIAA. 2010-9101.

[20]Forrester, A. I., and Jones, D. R., "Global Optimization of Deceptive Functions with Sparse Sampling". 2nd AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008, Paper AIAA 2008-5996.

[21]Forrester, A. I., Sóbester, A., and Keane, A. J., "Optimization With Missing Data". Proceedings of The Royal Society A, Vol. 462, Januray 2006, pp. 935–945.