AIAA AVIATION Forum
August 2-6, 2021, VIRTUAL EVENT
AIAA AVIATION 2021 FORUM

10.2514/6.2021-3078

System Architecture Optimization: An Open Source Multidisciplinary Aircraft Jet Engine Architecting Problem

J.H. Bussemaker*
DLR (German Aerospace Center), Institute of System Architectures in Aeronautics, Hamburg, Germany

T. De Smedt†, G. La Rocca‡
Delft University of Technology, Faculty of Aerospace Engineering, Delft, The Netherlands

P.D. Ciampa§, B. Nagel¶
DLR (German Aerospace Center), Institute of System Architectures in Aeronautics, Hamburg, Germany

Decisions regarding the system architecture are important and taken early in the design process, however suffer from large design spaces and expert bias. Systematic design space exploration techniques, like optimization, can be applied to system architecting. Realistic engineering benchmark problems are needed to enable development of optimization algorithms that can successfully solve these black-box, hierarchical, mixed-discrete, multi-objective architecture optimization problems. Such benchmark problems support the development of more capable optimization algorithms, more suitable methods for modeling system architecture design space, and educating engineers and other stakeholders on system architecture optimization in general. In this paper, an engine architecting benchmark problem is presented that exhibits all this behavior and is based on the open-source simulation tools pyCycle and OpenMDAO. Next to thermodynamic cycle analysis, the proposed benchmark problem includes modules for the estimation of engine weight, length, diameter, noise and NOx emissions. The problem is defined using modular interfaces, allowing to tune the complexity of the problem, by varying the number of design variables, objectives and constraints. The benchmark problem is validated by comparing to pyCycle example cases and existing engine performance data, and demonstrated using both a simple and a realistic problem formulation, solved using the multi-objective NSGA-II algorithm. It is shown that realistic results can be obtained, even though the design space is subject to hidden constraints due to the engine evaluation not converging for all design points.

Nomenclature

API	=	Application Programming Interface	MBSE	=	Model-Based System Engineering
BPR	=	Bypass Ratio	MDO	=	Multidisciplinary Design Optimization
CRTF	=	Counter-Rotating Turbofan	OPR	=	Overall Pressure Ratio
ECS	=	Environmental Control System	TIT	=	Turbine Inlet Temperature
FAR	=	Fuel-to-Air Ratio	TSFC	=	Thrust-Specific Fuel Consumption
FPR	=	Fan Pressure Ratio	UHBR	=	Ultra-High Bypass Ratio
GTF	=	Geared Turbofan	XDSM	=	eXtended Design Structure Matrix
ITB	=	Inter-Turbine Burner			

*Researcher, MDO group, Aircraft Design & System Integration, Hamburg, jasper.bussemaker@dlr.de
†M.Sc. Thesis student, Flight Performance and Propulsion, Delft, thibault.desmedt@me.com
‡Associate professor, Flight Performance and Propulsion, Delft, g.larocca@tudelft.nl, AIAA member
§Head of MDO Group, Institute of System Architectures in Aeronautics, Aircraft Design & System Integration, Hamburg, pier.ciampa@dlr.de, AIAA MDO TC member
¶Institute director, Institute of System Architectures in Aeronautics, Hamburg, bjoern.nagel@dlr.de

I. Introduction

DECISIONS taken early on in the design process of complex systems greatly impact the performance of the final design. Whilst it is important to have a good understanding of the impacts of these decisions, in practice this is difficult to achieve, as during the early design stage little is known about the behavior of the system. Additionally, system design spaces are often extremely large due to the combinatorial explosion of alternatives. To solve these problems, systematic design space exploration techniques need to be applied during the design of a complex system, also at the earlier design stage when the architecture of the system has not been fixed yet [1]. This would increase the understanding of the impact of important architectural decisions, enable the effective exploration of the design space, and reduce reliance on expert judgment suffering from bias, subjectivity, conservatism or overconfidence [2].

At the DLR and TU Delft, collaborative efforts are undertaken to improve the design process of complex systems, by extending our collaborative Multidisciplinary Design Optimization (MDO) capabilities to also cover the optimization of system architectures [3]. The goal is to bridge upstream Model-Based System Engineering (MBSE) processes, dealing mostly with scenarios, stakeholders, needs, goals, and requirements identification, and downstream MDO processes, dealing mostly with detailed multidisciplinary engineering analyses and design space exploration. The best system architecture, or rather the Pareto front of optimal system architectures, can then be identified by applying formal design optimization techniques to the system architecture design space.

Several factors contribute to the difficulty of solving system architecture optimization problems. One, common to any collaborative MDO pursue, is that the performance analysis model generally consists of an expensive black-box function. This means that no a-priori information is available about the shape and topology of the design space, and that the number of function evaluations needed for finding the optimal designs should be as low as possible. Next, as shown in [1], the architecture optimization problem is a mixed-discrete, multi-objective, hierarchical problem: *mixed-discrete* because design decisions might consist of both categorical architecting decisions and integer or continuous sizing parameters, *multi-objective* because the evaluation of possible architectural choices is generally based on multiple conflicting stakeholder selection criteria, and *hierarchical* because design variables can be conditionally active based on other design variables. In fact, the hierarchical nature is generally a consequence of the aforementioned categorical variables, as the presence of certain component(s) in the system architecture comes with the necessary design variables to define said component(s).

These characteristics make for a class of challenging optimization problems. It is expected that existing optimization algorithms will have difficulties with either solving the problem at all (i.e. finding an optimal architecture), or with solving the problem without needing an excessive number of function evaluations. In light of this, it is important that existing and new optimization algorithms are developed for and tested on realistic architecture optimization problems. Thorough literature research showed that currently no such benchmark architecture optimization problem exists. This work aims at filling said gap with the development of a benchmark optimization problem based on a realistic aircraft jet engine architecting framework, for the specific purposes of:

1) Supporting the development, evaluation, and comparison of optimization algorithms suitable for system architecture optimization;
2) Supporting the development, evaluation, and comparison of system architecture design space modeling methods;
3) Educating researchers, engineers, and other stakeholders on the behavior and characteristics of system architecture optimization problems.

II. Benchmark Problem Definition

A. Architecture Design Space

One step of the systems engineering process is defining the system architecture [4]. In systems engineering, the product to be developed is seen as a system: a set of interconnected components, that together perform more than simply the sum of the components [5]. To make sure that all stakeholder needs and requirements are being met, *what* the system does is defined before determining *how* the system does it. The system functions describe what the function does, and thereby represent the purpose of the system. The system form is what will be implemented in the end, and the elements of form (also called system components) exist to fulfill the system functions. A system architecture then describes the allocation of function to form, and the relationships among the elements of form: it is a function-form-structure mapping.

The system architecting process itself can be seen as a decision-making process. Many design decisions are usually identified, and the best option for each decision needs to be determined. To formalize the architecting process into

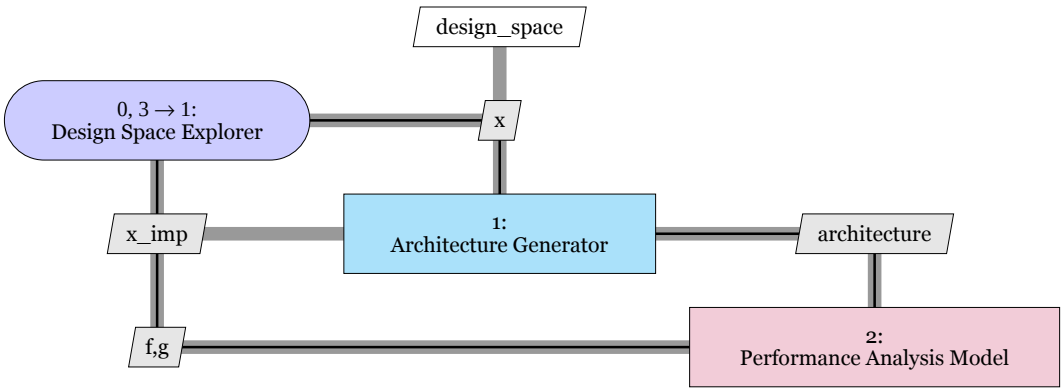


Figure 1 XDSM [7] view of the coupling between the design space explorer and the analysis response model for an architecture optimization problem. The architecture generator step uses the design space model to interpret the design vector x into an architecture instance, which is then used by the analysis model to estimate the architecture performance. Due to design variable hierarchy and the possibility of infeasible architectures, the design vector can be modified (imputed) at this stage, which needs to be communicated back to the design space exploration algorithm. The design space explorer can then use this information to suggest a new design vector. Fig. from [1].

an optimization problem, which would enable systematic exploration of the design space, the design space needs to be modeled in such a way that decisions are clearly identified, so that architecture candidates can be automatically generated, and that each architecture candidate correctly represents the function-form-structure mapping. Architecting decisions are usually of a categorical nature, consisting of a set of mutually exclusive decision options. However, in general also integer or continuous sizing parameters might be included as architecture-level decisions. Examples of these different types of decisions would be the integration of the engine in the airframe (wing- or fuselage mounted) for categorical decisions, the number of engines on an aircraft for integer parameters and the booster diameter in the rocket architecting problem presented in [6] for continuous parameters.

Another special characteristic of architecture design spaces is decision hierarchy [1]: some architecting decisions can themselves be active or not based on other architecting decisions. For example, if one decision is to choose whether to use a canard or an aft-fuselage empennage for pitch control of an aircraft, and another decision is to choose the aft-fuselage empennage layout (e.g. conventional, T-tail, V-tail), then the second decision is not active if a canard has been chosen. This effect gives rise to the distinction between the *apparent* architecture design space, which spans the combinations of all design variables, and the *feasible* architecture design space, which only represents the combinations of design variables that represent valid system architectures. For compatibility with existing optimization algorithms, it is therefore needed to include an additional architecture generation step in the conventional optimization loop, to make sure that only design vectors representing valid system architectures are analyzed. The process of correcting a design vector to represent a valid architecture design point is called imputation [8]. An example would be to disable the aft-fuselage empennage layout decision when a canard configuration is selected, as discussed before. Additionally, the architecture generation step enables the representation of the system architecture in a more problem-appropriate format than a design vector due to the creation of an actual architecture, which is more convenient to inspect. This concept is visualized in Fig. 1.

Design optimization involves quantitatively comparing the performance of the generated design candidates. To do this, performance metrics that apply to all architectures, regardless of their structure and included functions and components, are defined. Performance metrics can be interpreted as objectives or as constraints. Additionally it must be noted that because of the many stakeholders involved in the design of a system, it is needed to create a ranking of system goals. In general it can be said that when designing a system, it should always fulfill several conflicting goals. Although goals can be interpreted both as objectives and as constraints in the context of optimization, in general it can be assumed that there will be multiple optimization objectives: the system architecture optimization problem is of a multi-objective nature. This means that there is not one optimal architecture, but rather a set of Pareto-optimal architectures: each of these architectures is better than any of the other architectures in at least one of the objectives, but worse in terms of at

least one other objective as well [9].

B. Benchmark Problem Requirements

When it comes to optimization benchmark problems, Gray [10] mentions they should represent realistic engineering challenges by appropriately mimicking the design space behavior and dimensionality, they should have multiple difficulty levels ranging from easy familiarization problems to high-fidelity problems, and they should be open-source and based on open-source tools to aid reproducability, research, and education.

From the considerations discussed in this section, it can be concluded that to create a realistic system architecture optimization benchmark problem, it should adhere to the following requirements:

R1 *Design variables should be mixed-discrete:* both categorical, integer and continuous variables must be included in the design vector, so that the optimization problem becomes a mixed-discrete problem.

R2 *Multiple conditionally active design variables should be included:* conditionally active design variables are active or not based on some other design variable choice, and thereby introduce a design variable hierarchy. This is a common occurrence in system architecture design spaces and should therefore be represented.

R3 *The design space should have between 10 and 50 dimensions:* an appropriately high design space dimensionality should be defined to make the problem realistic enough. For system architecting problems it is expected that up to several tens of design variables can be present.

R4 *There should be multiple objectives:* there should be approximately two to ten non-correlated design objectives to choose from, representing conflicting stakeholder needs.

R5 *The performance evaluation function behavior should be black-box:* nothing is known a-priori of the shape and behavior of the design space due to the black-box nature of the evaluation function, and in general the design space should be assumed to be nonlinear.

R6 *The difficulty of the problem should be tunable:* different difficulty levels serve different user needs: an easy problem serves as a familiarization with the problem and with system architecting in general, more difficult levels serve as realistic test cases for optimizer performance.

R7 *The problem should be open-source itself and be based on open-source tools:* this helps in reproducing and verifying study results, and makes it accessible to a wider audience for research and education purposes.

R8 *The problem should be hackable:* it should be easy to modify the problem structure and to include additional elements, like engineering disciplines, design decisions, objectives, and constraints.

C. Aircraft Jet Engine Design

The benchmark problem presented in this paper will be that of the design of an aircraft jet engine for a given aircraft mission. This choice stems from:

- The strongly multidisciplinary nature of engine design (e.g. thermodynamic cycle analysis, mechanical and aerodynamic design of turbomachinery, airframe integration);
- The interesting and non-trivial architecting choices present in aircraft engine design problems (e.g. turbojet or turbofan, number of shafts);
- The inclusion of categorical decisions (e.g. are certain technologies included or not) and integer or continuous parameters (e.g. number of shafts, bypass ratio, pressure ratios), in which the number of decisions can easily be varied while still resulting in realistic engine designs;
- The presence of decision hierarchy (e.g. if there is no fan, there is no bypass ratio variable);
- The recent availability of open-source multidisciplinary engine design tools;
- The multiple conflicting stakeholder requirements (e.g. thrust, fuel consumption, noise, emissions) which can be used as nonlinear black-box objectives for the optimization, depending on which engineering disciplines are available;
- The general experience with overall aircraft design of such aircraft sizes at the organizations of the authors.

This makes the aircraft engine design problem very suitable to be used as a realistic benchmark problem for system architecture optimization.

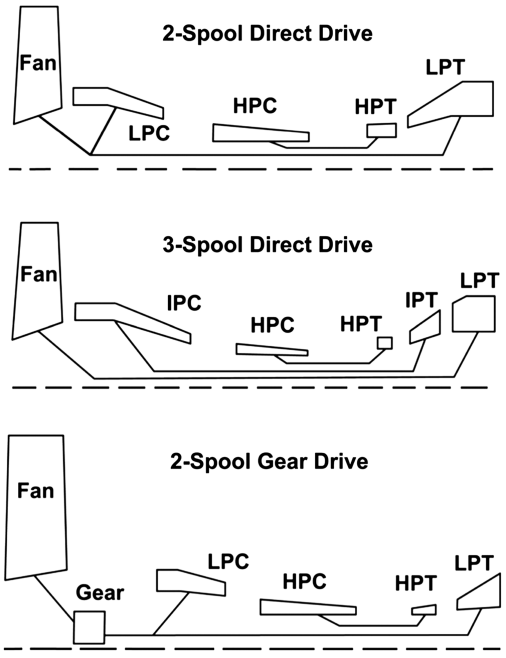


Figure 2 Example of turbofan architectures, from [11].

Central to the design of gas turbine engines is the thermodynamic cycle analysis [12]: the estimation of engine performance (e.g. thrust, fuel consumption, emissions), while taking design limits (e.g. max turbine temperature), flight conditions (e.g. Mach number, altitude), and design choices (e.g. pressure ratios, bypass ratio) into account. Cycle analysis relates the performance of all engine components to each other, and thereby acts as the engine-system-level analysis that integrates results of detailed components level analysis (e.g. high-fidelity CFD analysis of the compressor, combustor, or turbine) to determine the system-level impact of the individual components.

Engine architecture decisions mostly relate to the structure of the thermodynamic cycle. For example, is a bypass duct included (turbofan vs turbojet)? How many compressor and turbine stages are included (see Fig. 2)? Is the fan directly connected to the low-pressure compressor, or is a gearbox inserted [13]? At what stage is air bled off for the environmental control system (if at all)? From which shaft is power extracted for generating electricity? If in addition more innovative engine concepts are considered, questions might also include [14, 15]: is an intercooler added and where in the cycle [16, 17] (see Fig. 3)? How many core cycles are used (e.g. dual cycle) [18]? Is a Counter-Rotating Fan (CRTF) included [19]? Can the fan nozzle be variable [20]? Can the fan pitch be varied according to flight condition [21]? Can a hybrid-electric configuration be used to boost the engine [22]? Is the engine core arrangement reversed [23]? Each of these questions relates to one or more architecture decision variables, and can only be answered by systematically exploring the coupled impact of each of these decisions. It is therefore important that the analysis toolchain correctly captures the salient effects of each of these technologies.

The proposed benchmark problem is implemented in Python 3 and based on an engine cycle analysis framework called pyCycle [24]. This open-source framework is the latest development in a long line of engine analysis platforms, allowing modular definition of engine cycles, and easy integration in multidisciplinary analysis and optimization toolchains. Additionally, the analysis code provides analytical derivatives for all parameters, greatly accelerating the design and optimization convergence speed for a single given engine cycle. The engine cycle framework is implemented using OpenMDAO as MDO integration framework. OpenMDAO is an open-source MDO framework based on Python, allowing the calculation of analytical derivatives for MDO-system-level parameters based on analytical derivatives defined at the discipline level [25]. The combination of pyCycle and OpenMDAO thus allows a modular definition, analysis, and optimization of engine cycles, and the easy integration of additional disciplines in addition to thermodynamic cycle analysis. Additional disciplines that are considered in aircraft engine design include component weight estimation [26], noise analysis [12], nacelle geometry and aerodynamics [27], and reliability and cost analysis [11].

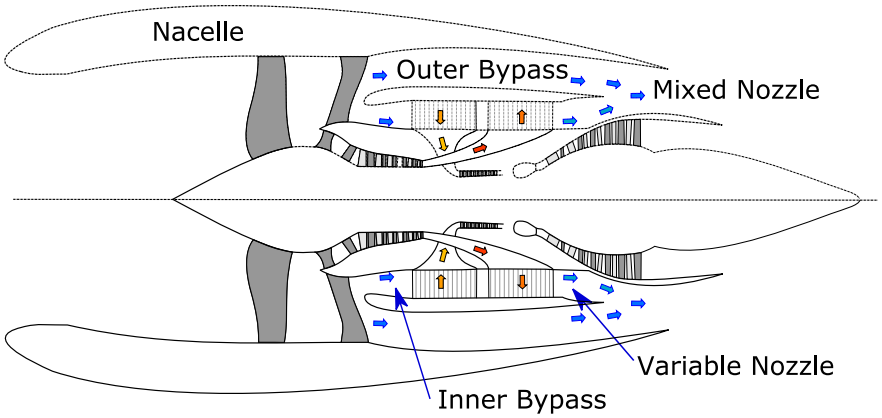


Figure 3 Engine architecture with intercooler to reduce the high-pressure compressor inlet temperature, from [16].

III. Benchmark Problem Implementation

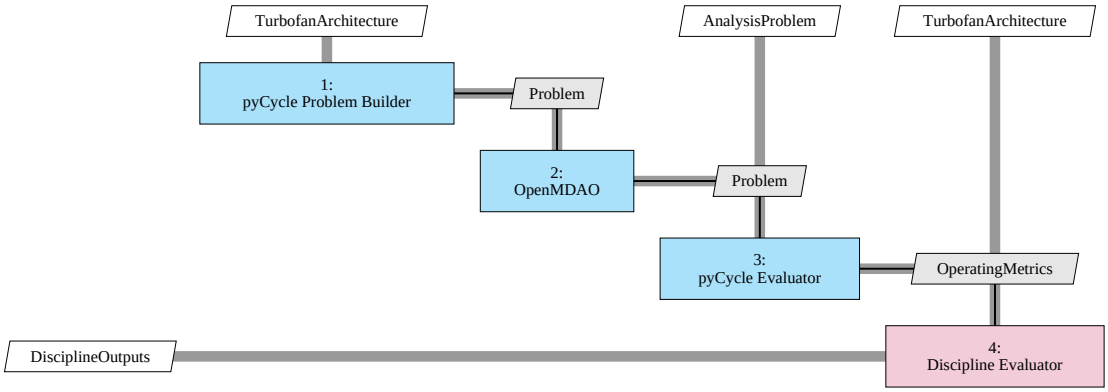
Looking at Fig. 1, it can be seen that setting up and executing a system architecting problem consists of several steps. First, there must be some way to model the design space and use this model to define the design vector, objectives, and constraints. Then, these need to be communicated to the exploration (i.e. optimization) algorithm. The exploration algorithm will then systematically vary the design vector and ask the performance analysis model to evaluate this design vector. To evaluate, the design vector is then first translated into a system architecture representation, making sure that any decision hierarchy is taken care of, and this architecture is then analyzed by the appropriate multidisciplinary analysis toolchain. To facilitate this process, the benchmark problem will consist of two main components (see Fig. 4 for a visual representation):

1) *The engine architecture evaluator:* this component provides a way to define and evaluate an engine architecture for specific operating conditions, by translating the architecture definition into a pyCycle problem. This component operates at the level of the individual architecture and is visualized in Fig. 4a.
2) *The engine architecting framework:* this component provides a way to describe the architectural choices, and from there defines a design vector, and functionality for translating a design vector into an engine architecture definition. This component operates at the level of the architecture design space and optimization problem and is visualized in Fig. 4b.

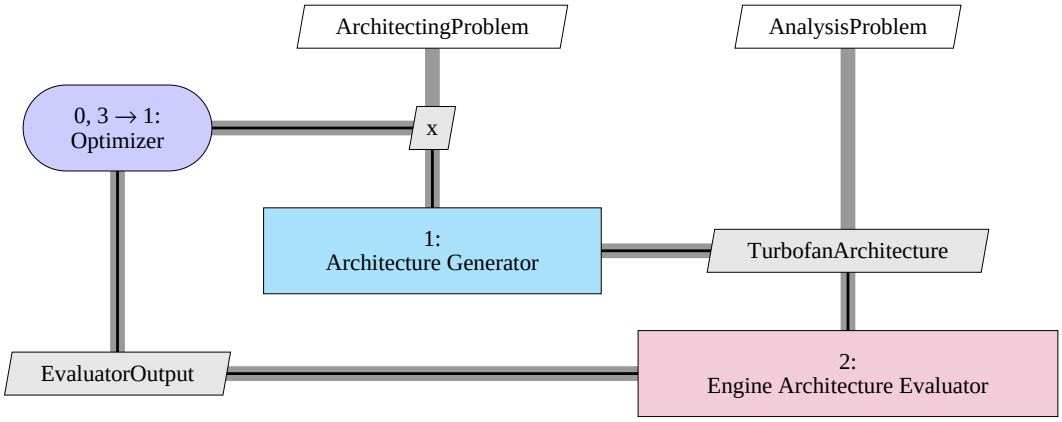
This separation is made so that it is clear that the evaluation of individual architectures is distinct from the definition of the architecture design space. Together, these components comprise a standalone engine architecting problem, but both the components can also be replaced for research purposes. In particular, this setup facilitates research into, and comparison of, methods for modeling system architecture design spaces and formulating architecture optimization problems.

The rest of this section presents and discusses the implementation of the two components. First, the engine architecture evaluator is discussed, then the engine architecting framework is discussed as it depends on features of the evaluator component. The code as presented in this section has been published at [28]*. Readers should be able to run the benchmark problems presented in this paper.

*jbussemaker/OpenTurbofanArchitecting



(a) XDSM of the "engine architecture evaluator" component of the benchmark problem: it defines the architecture definition format, translates architecture definitions into a pyCycle problem, and executes OpenMDAO to extract the desired metrics (objectives and constraints).



(b) XDSM of the "engine architecting framework" component of the benchmark problem: it defines an architecture problem, constructs the design vector from this definition, and translates design vectors to an engine architecture definition to be evaluated.

Figure 4 Overview of the two benchmark problem components, which can be combined by replacing the "Engine Architecture Evaluator" block in Fig. 4b by Fig. 4a.

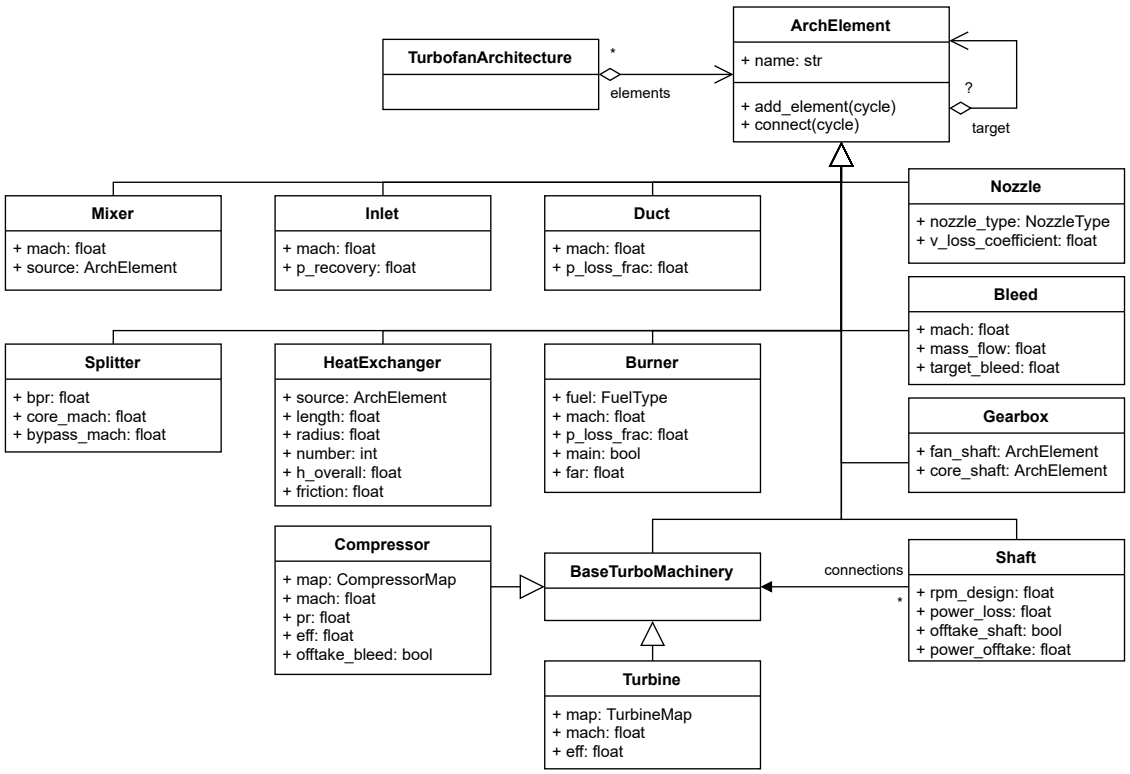


Figure 5 Engine architecture class diagram: TURBOFANARCHITECTURE is an aggregation of one or more ARCHELEMENTS. All components are derived from pyCycle, except the HeatExchanger which was developed by the authors.

A. The Engine Architecture Evaluator

The purpose of the engine architecture evaluator is to perform design and simulation of a given engine architecture, see Fig. 4a. It does this by building an OpenMDAO problem using pyCycle and any additional disciplines that might be needed to evaluate the requested metrics. This section introduces the engine architecture format that is used to define architectures, the setup of the thermodynamic cycle analysis using pyCycle, and the aircraft engine discipline evaluator.

1. Engine Architecture Definition

The engine architecture is defined according to the class diagram shown in Fig. 5. The architecture is defined as instances of Python objects, and it is trivial to develop an additional interface so that these objects are instantiated from the definition in some text file format (e.g. XML, JSON) so that the architecture evaluation tool could be implemented as a standalone tool. The engine architecture is defined by architecture elements that are connected to each other. The kind of elements defined in this diagram (e.g. inlet, compressor, burner, turbine) are common across most thermodynamic cycle analysis frameworks, and are based on the components implemented by pyCycle itself [24]. The only addition of the authors is the heat exchanger component, to model architectures wtth an intercooler. In order to determine net thrust, each architecture should have one inlet and at least one nozzle, which are the elements connected to the freestream flow.

To evaluate an engine architecture, operating conditions need to be specified (see Fig. 6): the Mach number, altitude, temperature difference compared to the International Standard Atmosphere (ISA), required thrust, and bleed and power offtake requirements. An engine cycle analysis always contains one set of design conditions: these are the operating conditions for which the engine cycle is sized, for example for a given Turbine Inlet Temperature (TIT). Then, there can be any number of evaluate conditions (or off-design conditions) where the engine is evaluated at, but which do not size any of the components. Additionally, each operating condition has a balancer. Balancers represent extra equations needed for solving the thermodynamic cycle using implicit state variables; this principle is well-known in the thermodynamic cycle analysis literature, see for example [24].

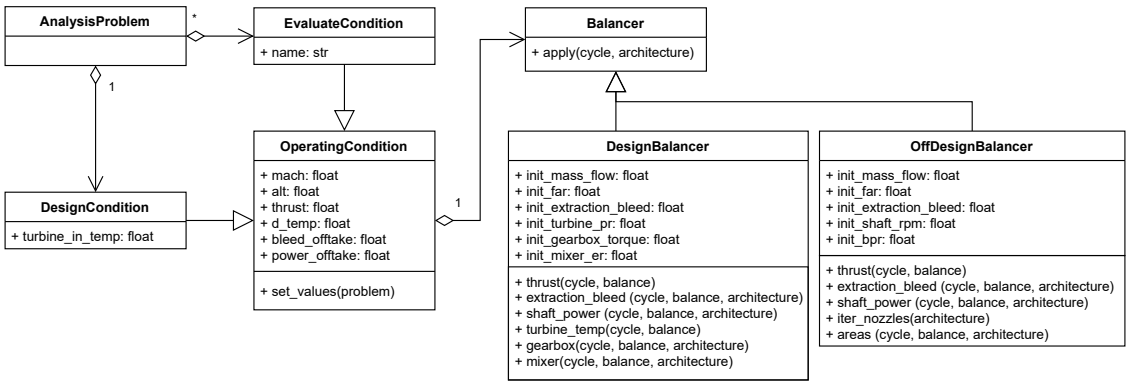


Figure 6 Definition of the analysis problem: the operating conditions and the balancers.

2. Thermodynamic Cycle Design: pyCycle

The core of the engine analysis consists of the thermodynamic cycle analysis done using pyCycle [24]. pyCycle is a library of engine components that can be combined to create an OpenMDAO problem that performs thermodynamic cycle analysis. A typical pyCycle problem consists of a multi-point pyCycle component, that contains multiple cycle components (one design cycle and multiple off-design cycles), each representing one operating condition. Quantities like sized cross-sectional areas are fed from the design cycle to the off-design cycles to ensure consistency. Each of the cycle components includes the full engine architecture as composed from the elements in the pyCycle library, and a balancer component with multiple balancing equations. As explained in [24], these balancers determine how important engine parameters like thrust and TIT are tuned. For each of the operating conditions performance parameters are additionally determined, for example Thrust-Specific Fuel Consumption (TSFC), fuel flow, thrust, and Overall Pressure Ratio (OPR).

The engine architecture evaluator constructs a pyCycle problem from a TURBOFANARCHITECTURE and an ANALYSISPROBLEM instance (see Section III.A.1). Each of the architecture elements contains the logic required for adding the element to the cycle, for connecting the element to the next element in the architecture (e.g. the compressor to the burner, or the splitter to the core and bypass flows), and for adjusting specified component parameters (such as efficiency, which is not a design variable). Then, the design point is connected to the off-design points, and the operating conditions are set. The result is an OpenMDAO problem that can be executed using the regular OpenMDAO API.

The thermodynamic cycle design thereby provides sizing of the engine and common performance parameters like OPR, fuel flow, thrust and TSFC. Additionally, flow properties (static and total) are available for all flows (input and output) of all elements. These pyCycle thermodynamic cycle results will also be used in the discipline evaluator.

3. Discipline Evaluator

Next to the thermodynamic cycle, there are several other disciplines that nned to be taken into account during the design of an aircraft engine such as weight, nacelle geometry, noise and emissions. Evaluation of all disciplines is done using open-source tools in order to comply with the requirements of Section II.A for creation of a realistic benchmark problem. These extra disciplines are calculated after the thermodynamic cycle analysis (see Fig. 4a), and all disciplines, except the NOx calculations, need both the output of the thermodynamic cycle analysis and the architecture definition as inputs. Calculation of NOx emissions only needs thermodynamic cycle calculation output.

Engine Weight Estimation One of the most important disciplines in aircraft engine design is weight. A weight estimation method has been implemented based on [29], which correlates bypass ratio, OPR, core mass flow rate and presence of a gearbox to the weight of the engine. Next to that, in order to compare engine architectures with e.g. different number of shafts or burners, correction factors were applied based on the engine component weights in [30] to correct for systems and other integration effects. To estimate the complete engine system weight, the nacelle weight is calculated with the method described in [31] and added to the bare engine weight. This discipline adds an interesting extra design objective that normally opposes TSFC: a more fuel-efficient engine might also be heavier, leading to a tradeoff for the designer of the engine. It might also be used as a constraint, which can be interpreted as a weight budget coming from overall aircraft design considerations.

Geometry Estimation To ensure that the jet engines can be integrated in the airframe without issues, the geometry of the nacelle needs to be taken into account. In the benchmark problem, the maximum length and diameter of the aircraft engine nacelle are estimated using a method from [32] which uses empirical correlations to size both turbojets and turbofans. Similarly to the engine weight estimation, the method uses engine parameters such as mass flow rate and BPR, and corrections were applied for engine architectures with e.g. multiple shafts or burners. The inclusion of geometry considerations adds another interesting design objective as larger engines might be more efficient, but also more difficult to integrate. Due to the limits in space under the wings, the nacelle geometry might be used as a constraint as well.

Environment Impact To estimate environment impact of the engine, noise and NOx emissions are calculated. The estimation of CO2 was not implemented in the benchmark problem as it is linearly dependent on fuel flow [33], which would lead to a conflict with Req. 4 of Section II.B. Noise and emissions are regulated by the International Civil Aviation Organization (ICAO) and other (inter)national aviation authorities. Again, empirical correlations are used to estimate the disciplines: [34] for noise using jet nozzle properties and atmospheric conditions, and [35] for NOx emissions using the combustion chamber inlet temperature and pressure. The environment considerations can be used both as objective and constraint in the optimization process.

B. The Engine Architecting Framework

The purpose of the engine architecting framework is to define the architecting problem and then use the engine architecture evaluator to find the best engine architecture. The implementation consists of two parts: the interface for defining the system architecting problem, and the architecture generator that translates design vectors as generated by the optimization algorithm to TurbofanArchitecture instances that can be evaluated by the engine architecture evaluator. See also Fig. 4b.

1. Architecting Problem Definition

The architecting problem defines the architecture choices and design metrics. Operating conditions are also defined here, so that the engine architecture evaluation component can be called directly without needing any additional interaction. The main use case of the architecting problem definition is to formally define an optimization problem for testing optimization algorithms. The engine architecting framework provides interfaces to the Python multi-objective optimization framework pymoo [36] to help with testing algorithms.

Choice Definition Choices semantically define architecting decisions to include in the architecting problem. One choice can map to one or more discrete or continuous design variables, and therefore adding choices increases the size of the architecture design space. If no choices are defined for the architecting problem, the resulting architecture represents a simple turbojet: one shaft, compressor, turbine, and burner that produces all of the thrust. Choices can be included as free or as fixed: the latter option might be used to modify this turbojet default.

The implemented architecting choices and their design variable mappings can be found in the list below, in order of execution. The order of execution is important as some architecting choices are dependent on decisions made in previous architecting choices.

1) The first architecting choice is the inclusion of a fan at the front of the engine. In case a fan is not included, the resulting engine architecture is a turbojet. Otherwise, the architecture is identified as a turbofan. For a turbofan architecture, two additional design variables are activated: the Bypass Ratio (BPR), with bounds between 2 and 12.5 and the Fan Pressure Ratio (FPR), with bounds between 1.1 and 1.8. These bounds are based on existing engines.
2) If the result of the first architecting choice is to include a fan, the option arises to include a second fan at the front of the engine which rotates at the same speed but the opposite direction as the main fan. This concept is called a CRTF. A CRTF architecture could result in a decrease of 5 EPNdB in noise, at the cost of 10% weight and 5.74% TSFC increase at cruise compared to an Ultra-High BPR (UHBR) engine [37].
3) Next, the number of shafts is decided: the engine architecture will have either one, two or three shafts. These can be referred to as the low-, intermediate- and high-pressure shafts. The advantage of multiple shafts is that the compressors and turbines can rotate at optimal speeds leading to a better engine performance. However, it also results in an increase in weight and complexity of the system. For each shaft, the RPM is a design variable. Other

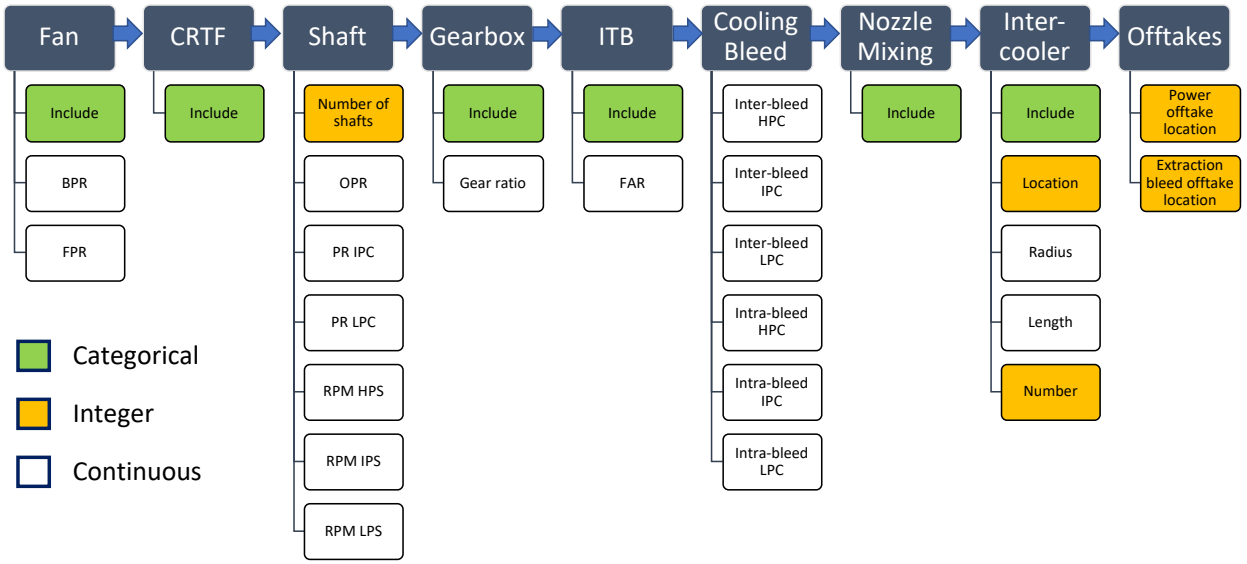


Figure 7 Visual representation of the engine architecting decisions determining the final engine architecture, including the different design variables which can be categorical, integer or continuous in nature. Arrows indicate order of execution.

design variables include the Overall Pressure Ratio (OPR) of the complete engine, as well as the percentage of the OPR that each compressor generates.

4) To enable the fan to rotate at an optimal speed, a gearbox can be inserted between the low-pressure shaft and the fan shaft. This architecture is referred to as the Geared Turbofan (GTF). The GTF option increases efficiency and decreases noise, due to the slower rotation speed of the fan, however comes at the price of possibly higher weight. In case a gearbox is included, the gear ratio is a design variable.
5) It is possible to include a second combustion chamber in the engine architecture, called an Inter-Turbine Burner (ITB). As the name suggest, the ITB is located between two turbines aft of the main combustion chamber. As the combustion process of the ITB occurs at high pressure, its efficiency is higher compared to for example an afterburner, reducing the fuel consumption. In addition, an ITB achieves lower NOx emissions and performs better in off-design conditions compared to a conventional turbofan. However, disadvantages include an increased weight and complexity of the system [38]. When the ITB is selected, its Fuel-to-Air Ratio (FAR) is a design variable.
6) In order to cool the turbine or to regulate the axial velocity of the compressor gases, air can be bled from the compressors through bleed valves. This air is referred to as cooling bleed and is a design choice, opposed to the extraction bleed (see choice 9). For the engine architecting problem, the amount of cooling bleed and its targets (i.e. defined turbines) need to be specified for each individual compressor. To limit the engine performance effect, it is advised to bleed air at early stages in the compressor(s) [39]. The implementation of cooling bleed is split up into intra-bleed and inter-bleed: the former defines a situation where air is bled from within a compressor, whereas for the latter it is bled from in between two compressors.
7) For turbofans, a choice can be made between a separate or mixed flow nozzle. In the mixed flow nozzle, the flow of the core and the bypass is mixed at the end of the engine and leaves the engine through one joint nozzle, whereas this is not the case for a separate flow nozzle. A mixed nozzle results in a slightly higher efficiency, but also higher weight and length of the engine, which is why the separate flow nozzle is currently more often used in commercial aviation [39].
8) On stationary and marine gas turbines, a heat rejection method is sometimes implemented: this is called intercooling. During this process, heat is removed from in between two compressors using a heat exchanger resulting in lower fuel consumption and emissions, at the cost of higher engine volume and vibration issues [17]. In the architecting problem, an intercooler can be added by specifying its location in the engine and its geometry (radius, length and number of pipes).
9) Power and extraction bleed offtakes are specified as part of the engine requirements, and are therefore always

present in an engine architecture. Power offtakes are satisfied by extracting electrical power from one of the engine shafts in order to power different systems onboard the aircraft. For extraction bleed, air is bled from one of the compressors to support for example the Environment Control System (ECS) or anti-icing systems of the aircraft [39]. The offtake location (both for power and extraction bleed) can be specified with the design variables.

In total, the design space includes 41 design variables: 6 categorical, 5 integer and 30 continuous. Only the discrete variables already account for approximately 1.3 million engine design points, which was found by taking the product of the number of options for all the different discrete design variables.

Objective and Constraint Selection Objectives are metrics to be either maximized or minimized; constraints are metrics where a lower or upper limit is placed on the value. Available metrics to be used as objectives or constrains include TSFC, weight, length, diameter, NOx emissions, noise, and jet Mach number. All of these are minimized if selected as objectives. In addition to these metrics, a number of constraints will always be present in the architecting problem to ensure the feasibility of the generated engine architectures, including CRTF efficiency and intercooler length as compared to the overall engine radius.

2. Architecture Generator

Once the engine architecting problem has been defined, the optimization loop can be started and the optimizer will start generating design points to be evaluated. Each design point is defined as a design vector that first needs to be converted into an architecture definition before it can be evaluated by the engine architecture evaluator. The architecture generator converts the design vector into a TURBOFANARCHITECTURE instance using the selected architecture choices, in which each choice implements the logic to construct its corresponding part of the architecture.

The architecture generator additionally contains two mechanisms to deal with design variable hierarchy: imputation and result caching. Imputation converts design variables into their canonical form so that every evaluated design variable maps to only one architecture. Whenever an inactive design variable is encountered, its value is reset to some default value: for example a 0 for a discrete variable and mid-bounds for a continuous variable. This approach only works if the optimization algorithm accepts that the design variable of a design point can be modified (i.e. there is a feedback connection), as discussed in [1]. Stochastic algorithms like evolutionary algorithms usually have no problems with this, but more deterministic algorithms might not support the modification of the design vector.

The other mechanism, result caching, makes sure that if the same architecture would be requested to be evaluated more than once in an optimization loop, for example because two different design vectors result in the same architecture due to imputation, the architecture is only evaluated once: the subsequent times the architecture is requested to be evaluated, the results of the previous evaluation are simply returned. This mechanism is used in the naive approach of ignoring design variable hierarchy, as discussed in Section II.A.

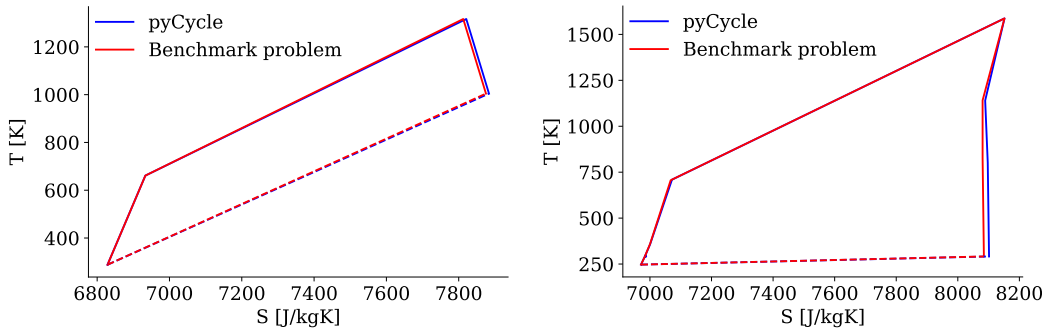
IV. Verification & Validation

The benchmark problem presented in this paper meets the general requirements for a system architecture optimization benchmark problem: it contains a sufficient amount of mixed-discrete and hierarchical design variables, multiple conflicting optimization objectives are available, evaluation is black-box from the optimizer perspective, it is easy to tune the difficulty of the problem by including more or less architecting choices, new features are easy to implement due to the object-oriented implementation of the problem, and the problem is released as open-source.

To show that it also represents a realistic engineering problem, this section presents a verification & validation and a demonstration of the benchmark problem. All verification & validation and demonstration results will also be included in the published package, so it can be independently verified by other parties.

A. Architecture Evaluation

In order to verify that the created choices in the benchmark problem lead to the same results as the code of pyCycle, a comparison of the thermodynamic cycle was made for two engine architectures: a simple turbojet with one shaft, and a high-BPR turbofan with two shafts and a fan. As can be seen in Fig. 8, the thermodynamic cycle of the benchmark problem and pyCycle is very similar which verifies that the architecting choices have been implemented correctly. The small discrepancies between both cycles are the result of pressure losses in ducts, which is a pyCycle component that was not included in the benchmark problem.



(a) Turbojet engine architecture, with one shaft.

(b) UHBR turbofan engine architecture, with two shafts and a fan.

Figure 8 Comparison of the thermodynamic cycle computed by pyCycle and the benchmark problem results for two different engine architectures, to verify whether the implemented design choices in the benchmark problem result in the same architecture as a given pyCycle example.

Table 1 Comparison between properties of the LEAP-1C engine and the high-BPR, two shaft engine validation architecture. Data from [40]. The TSFC is calculated at cruise condition whereas the other disciplines are calculated at take-off condition.

	TSFC [g/kNs]	Weight [kg]	Length [m]	Diameter [m]	NOx [g/kg fuel]	Noise [dB]
LEAP-1C	14.4	3932	4.51	2.71	18.77-64.36	Unknown
Validation	16.6	3910	4.49	2.71	61.99	121

Table 2 Comparison between properties of the F100 engine and the low-BPR, two shaft engine validation architecture. Data from [41]. The TSFC is calculated at cruise condition whereas the other disciplines are calculated at take-off condition.

	TSFC [g/kNs]	Weight [kg]	Length [m]	Diameter [m]	NOx [g/kg fuel]	Noise [dB]
F100	Unknown	1735	4.85	1.18	Unknown	Unknown
Validation	26.7	1752	4.82	1.35	44.38	108.29

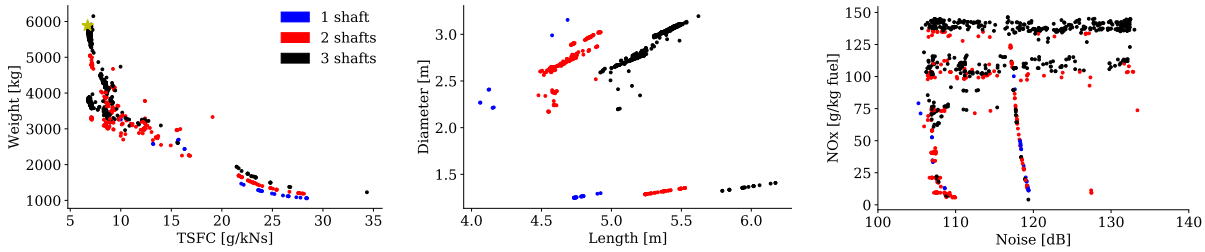
To validate the results of the benchmark problem disciplines, they were compared to existing engine architectures. For the turbofan, the CFM LEAP-1C engine is chosen whereas for the turbojet, the Pratt & Whitney F100 engine is selected. The TSFC of the engines is calculated at the cruise condition (35000 ft altitude at Mach = 0.8), whereas the other disciplines are calculated at take-off condition (0 ft altitude at Mach $\approx$ 0). Data was retrieved from [40] and [41]. In order to take into account the technological progress that has been achieved in the design of aircraft engines, correcting factors were applied to the length and diameter disciplines in order to bring them up to date. As can be seen in Tab. 1 and 2, the resulting values from the benchmark problem are within the range of expected values for the LEAP-1C and F100 engines. This validates the implementation of the benchmark problem for the turbofan engine.

B. Simple Single-Objective Architecting Problem

Next, the engine architecting framework was verified using a simple architecting problem. This architecting problem features only one objective (TSFC), and only design feasibility constraints. The included design choices are: fan inclusion (1), number of shafts (3), gearbox implementation (4), nozzle type (7), and offtake locations (9). The numbers indicate the specific architecting choices listed in Section III.B.1 which contain all necessary information for the reader. These design choices result in 15 design variables and the creation of solely conventional engine architectures, i.e. excluding CRTF, ITB and intercooler. In total, 15 distinct engine architectures and 216 engine design points can be generated taking into account the discrete design variables. The design condition is similar to the LEAP-1C engine:

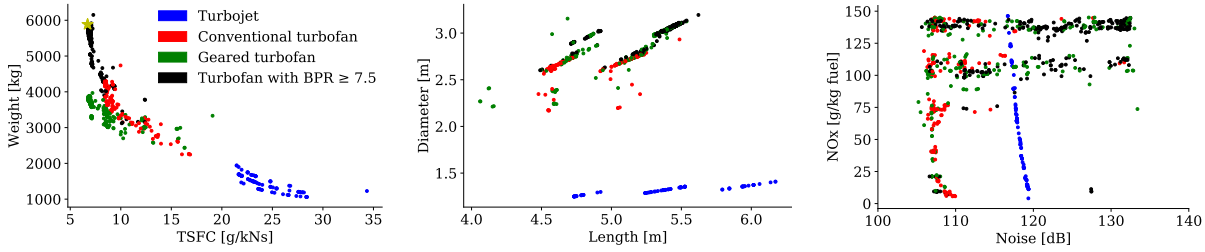
Mach ≈ 0, 0 ft altitude, 150 kN thrust, 1450 degC TIT, 0.5 kg/s extraction bleed, and 37.5 kW power offtake [42]. The multi-objective optimization algorithm NSGA-II [43] from the pymoo framework with a population size of 75 and termination criteria of 1000 evaluations was used to run the optimization.

All architectures were analyzed, and their results are presented in Fig. 9 and 10. All feasibility constraints, discussed in Section III.B.1, are satisfied. Approximately 49% of the generated engine architectures converged during the initial DOE, whereas this increased to 92% for the last iteration. This can be seen as the presence of hidden constraints [44]: constraints that are not known a-priori, and are considered violated whenever the simulation did not converge to a meaningful result. Hidden constraints are present in many simulation-based optimization problems, and can be challenging to deal with for the optimizer. NSGA-II is robust with respect to hidden constraints, as there the so-called extreme barrier approach [44] is applied: invalid results are assigned the value $+\infty$, to guide the selection and non-dominated sorting steps towards valid results.



(a) TSFC and weight. (b) Length and diameter. (c) Noise and NOx emissions.

Figure 9 Engine discipline results for the simple architecting problem, based on number of shafts.



(a) TSFC and weight. (b) Length and diameter. (c) Noise and NOx emissions.

Figure 10 Engine discipline results for the simple architecting problem, based on engine type. Note that the black dots represent architectures that do not include a gearbox.

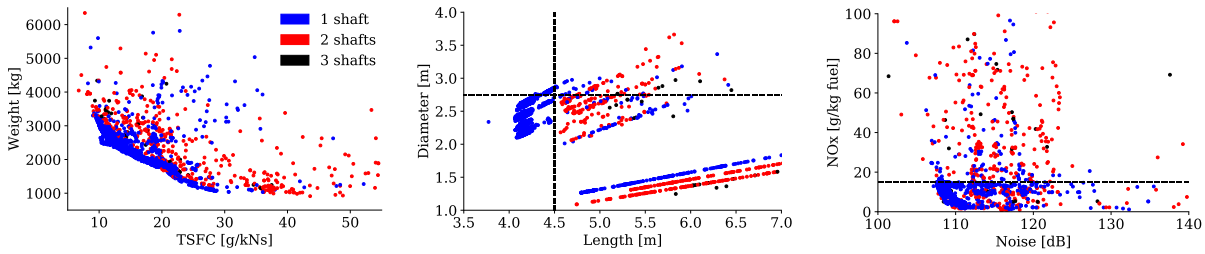
It can be seen that turbojets have higher TSFC and lower weight and lower diameter compared to turbofan engines, which is expected. For the number of shafts, it can be deduced that TSFC reduces whereas length and weight increase with increasing number of shafts. This could be explained by the fact that the additional compressors rotate at optimal speeds, leading to a higher efficiency of the engine at the cost of higher length and weight. The same principles apply to the geared turbofans: they seem to result in close to the lowest TSFC of all engine architectures, as compressors and fan rotate at optimal speeds, while simultaneously also providing low weight for the turbofan cluster. UHBR turbofans achieve a reduction in TSFC, but this comes at the price of higher weight due to the increased diameter of the engine [13]. The lowest TSFC in the entire optimization problem was also achieved by a UHBR turbofan at 6.7 g/kNs, and is indicated in Fig. 9a and 10a by a yellow star. Regarding NOx emissions, these increase with increasing engine OPR [45]. As a higher OPR can be achieved with a higher number of shafts, NOx emissions will increase with higher number of shafts as well. This can clearly be seen in Fig. 9. As noise is dependent on jet velocity and nozzle exit area, no significant deduction can be drawn for noise dependency on number of shafts or engine type [34].

Based on Fig. 9 and 10, the results of the simple architecting problem are in line with what was physically expected, serving as validation of the benchmark engine simple architecture optimization problem.

C. Realistic Multi-Objective Architecting Problem

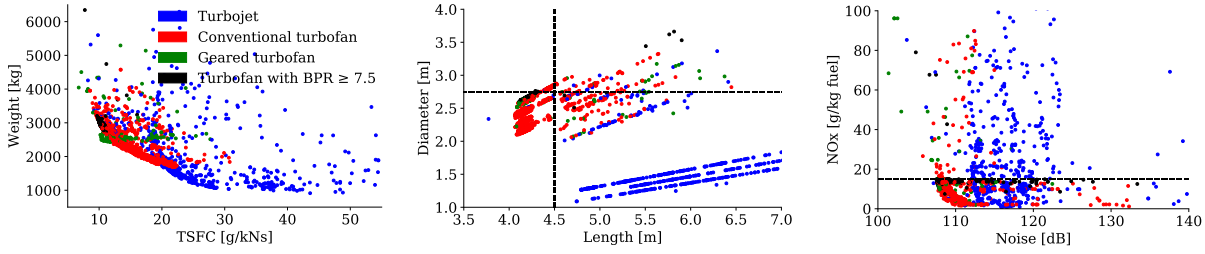
Finally, a realistic, more complex engine architecting problem is constructed and solved to provide a target Pareto front for future optimization algorithm research. The realistic architecting problem contains three conflicting objectives (TSFC, weight and noise), and multiple constraints (e.g. geometry and emissions). All available decisions discussed in Section III.B.1 are included in the problem, leading to 41 design variables. This results in the possibility to generate 85 distinct engine architectures and approximately 1.3 million engine design points taking all discrete variables into account. Evaluating one engine design point can take up to two minutes, making for a challenging optimization problem. The same design conditions and NSGA-II algorithm [43] are used as for the simple architecting problem. The population size was set to 205 and the termination criteria to 4000 evaluations, to account for the increase in number of design variables.

Results of the optimization are shown in Fig. 11 to 15. Compared to the simple architecting problem, it can be seen that the discipline results of the realistic architecting problem are in the same orders of magnitude and show approximately the same trends. Only the NOx emissions are significantly lower in the realistic architecting problem; this is due to the fact that NOx emissions of 15 g/kg fuel was used as a constraint. Furthermore, Fig. 14 shows that a Pareto front with non-dominated aircraft engine architectures has been formed. Additionally, Fig. 12 shows that two separate Pareto fronts can be distinguished: one for turbojets and one for turbofans, which is to be expected. Next to that, all constraints imposed on the architecture are satisfied. This shows that the benchmark problem achieves feasible results for the engine disciplines. From the generated engine architectures, 33% converged during the initial DOE whereas this increased to 92% for the last iteration. Similar to the simple problem, this demonstrates that the problem is subject to hidden constraints [44], and that NSGA-II is able to deal with these effectively.



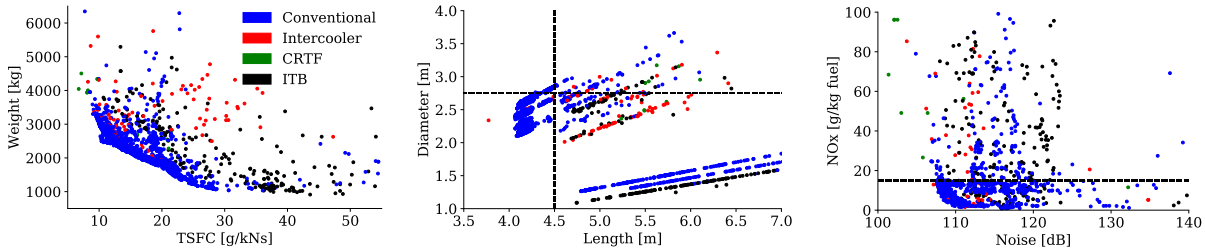
(a) TSFC and weight. (b) Length and diameter. (c) Noise and NOx emissions.

Figure 11 Engine discipline results for the realistic architecting problem, based on number of shafts. The black dotted lines are constraints to be minimized.



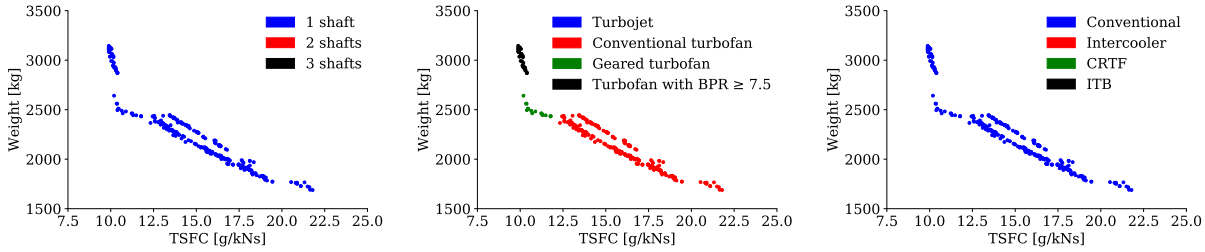
(a) TSFC and weight. (b) Length and diameter. (c) Noise and NOx emissions.

Figure 12 Engine discipline results for the realistic architecting problem, based on engine type. Note that the black dots represent architectures that do not include a gearbox and that the black dotted lines are constraints to be minimized.



(a) TSFC and weight. (b) Length and diameter. (c) Noise and NOx emissions.

Figure 13 Engine discipline results for the realistic architecting problem, based on innovative configurations. The black dotted lines are constraints to be minimized.

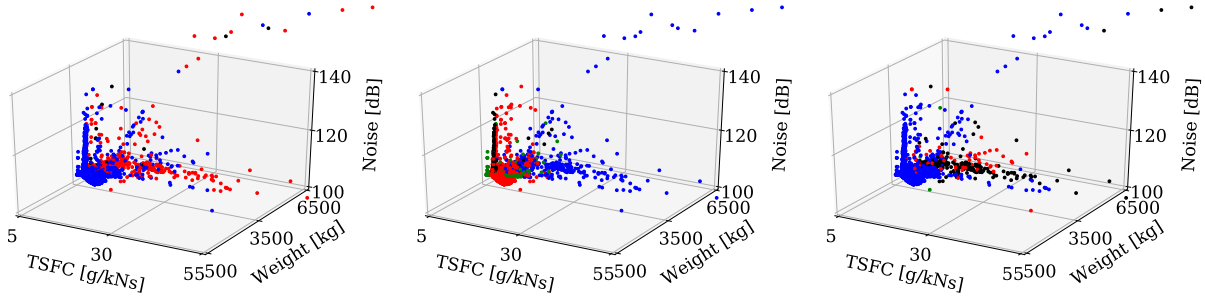


(a) Pareto front based on number of shafts. (b) Pareto front based on engine type. (c) Pareto front based on innovative configurations.

Figure 14 Pareto front of TSFC and weight for the realistic engine architecting problem. The color indications for each graph are indicated in Figures 11 to 13.

The results indicate that the optimizer has a preference for single shaft conventional turbofans. This can be explained by the fact that aircraft engines with only 1 shaft tend to have a lower weight as discussed in Section IV.B. It must be noted that this could be the result of assumptions in empirical correlation corrections, meaning that further research needs to be performed on the exact influence of shaft number on weight. Fig. 13a clearly indicates that 93% of the generated engine architectures are conventional architectures, which is in line with expectations:

- A CRTF usually increases TSFC and weight whereas only reducing noise slightly [37];
- An ITB reduces TSFC, however also increases engine weight significantly [38];
- An intercooler reduces both TSFC and NOx emissions, however intercooler implementation convergence issues were encountered during the optimization [17].



(a) Objectives based on number of shafts. (b) Objectives based on engine type. (c) Objectives based on innovative configurations.

Figure 15 Engine discipline results for the 3 optimization objectives of the realistic architecting problem: TSFC, weight and noise. The color indications for each graph are indicated in Figures 11 to 13.

V. Conclusions and Outlook

A realistic system architecture optimization benchmark based on an aircraft jet engine architecting problem was presented. System architecture optimization problems generally are hierarchical, mixed-integer, multi-objective, black-box optimization problems, which makes them difficult to solve. Benchmark problem requirements are defined with the aim to aid research into optimization algorithms capable of solving system architecture optimization problems, research into modeling methods for system architecture design space, and education on system architecture optimization in general.

An engine architecting benchmark problem was developed that uses pyCycle and OpenMDAO to analyze the thermodynamic cycles of a wide array of aircraft gas turbine engine architectures. Next to the thermodynamic cycle, other disciplines such as engine weight and noise were also implemented. The architecting problem allows the inclusion of many architecting choices to dynamically tune the difficulty of the architecting problem to solve. Several objectives and constraints can be selected to guide the design space exploration. The implementation of the benchmark problem consists of two components: the engine architecture evaluator and the engine architecting framework. The architecture evaluator analyzes one specific architecture using pyCycle and OpenMDAO, whereas the engine architecting framework enables the definition of the architecting problem and the translation of design vectors into valid architecture definitions. The benchmark problem is verified by comparing engine analysis results with pyCycle example problems, and by running two engine architecting problems of different complexity level. The code implementing the benchmark problem presented in this paper is available at [28].

With the different design choices implemented in the benchmark problem, a total of 85 distinct engine architectures and approximately 1.3 million engine design points can be generated taking all discrete variables into account. Verification and validation showed that the discipline results of the benchmark problem were very similar to actual engine data of the CFM LEAP-1C and the Pratt & Whitney F100 engines. The optimizer managed to find multiple optimal aircraft engine architectures, ultimately forming a Pareto front, which satisfied all implemented discipline and feasibility constraints. Furthermore, the results of both the simple and realistic engine architecting problem showed trends which were in line with what was physically expected, such as decreasing TSFC and increasing weight for a higher number of shafts in the engine, serving as additional validation of the benchmark problem. In general, the optimizer showed a preference for conventional turbofan engine architectures.

It is expected that this benchmark problem provides a realistic engineering architecting problem to test optimization algorithms on. The analysis time of one engine architecture should not be prohibitive, such that also relatively inefficient algorithms can be fairly compared to more suitable algorithms. Due to the easy definition of the architecting problem and the release as an open-source package, setting up experiments for testing optimization algorithms should be trivial to do. The Pareto front of the realistic architecting problem can be used to compare results of developed optimization algorithms.

Both the simple and realistic architecture problem are subject to non-convergence issues in their design spaces: in the initial DOE only 49% and 33% of design points result in a converged simulation, respectively. This can be seen as the presence of hidden constraints: constraints that are not known a-priori, and exhibit themselves through invalid evaluation results. Hidden constraints can be challenging to manage, however are a reality in many simulation-based optimization problems.

The combination of the engine architecture evaluator and the engine architecting framework should give a good introduction to how an architecture optimization problem is implemented in practice. Especially that there needs to be some consideration on how to define and analyze individual system architectures, how to specify the architecture optimization problem (i.e. its design vector, objectives, and constraints), and how to translate between design vector (as generated by the optimization algorithm) and the architecture definition (as needed for analysis). The engine architecture evaluator additionally provides a good example for how multidisciplinary analysis toolchains should be implemented for a system architecture optimization. The benchmark problem will help educating disciplinary experts, analysis tool developers, system architects, system engineers, project managers, and engineering students on all the relevant aspects of architecture optimization problems.

Currently, the provided engine architecting framework does not provide any interfaces to upstream MBSE processes, like stakeholder identification and requirements definition. Because of the modular nature of the benchmark problem, the engine architecting framework can be replaced by a generally applicable method for modeling architecture design spaces, while still being able to use the engine architecture evaluator to analyze specific engine architecture instances. Future research into the best way of integrating system architecture optimization with the MBSE process is needed. Next to that, it is also advised to perform more research in the implementation of the engine disciplines to improve the validity of the discipline results.

Acknowledgments

The research presented in this paper has been performed in the framework of the AGILE 4.0 project (Towards Cyber-physical Collaborative Aircraft Development) and has received funding from the European Union Horizon 2020 Programme under grant agreement n° 815122.

References

[1] Bussemaker, J. H., Ciampa, P. D., and Nagel, B., "System Architecture Design Space Exploration: An Approach to Modeling and Optimization," *AIAA AVIATION 2020 FORUM*, American Institute of Aeronautics and Astronautics, 2020. doi:10.2514/6.2020-3172.

[2] Roelofs, M., and Vos, R., "Correction: Uncertainty-Based Design Optimization and Technology Evaluation: A Review," *2018 AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2018, pp. 1–21. doi:10.2514/6.2018-2029.c1.

[3] Ciampa, P., La Rocca, G., and Nagel, B., "A MBSE Approach to MDAO Systems for the Development of Complex Products," *AIAA Aviation Forum*, American Institute of Aeronautics and Astronautics, Reno, Nevada, 2020. doi:10.2514/6.2020-3150.

[4] Crawley, E., Cameron, B., and Selva, D., *System architecture: strategy and product development for complex systems*, Pearson Education, 2015. doi:10.1007/978-1-4020-4399-4.

[5] NASA, "NASA Systems Engineering Handbook," Tech. Rep. Rev 2, NASA, 2016.

[6] Aliakbargolkar, A., Crawley, E., Wicht, A., Battat, J., and Calandrelli, E., "Systems Architecting Methodology for Space Transportation Infrastructure," *Journal of Spacecraft and Rockets*, Vol. 50, No. 3, 2013, pp. 579–590. doi:10.2514/1.A32320.

[7] Lambe, A. B., and Martins, J. R., "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes," *Structural and Multidisciplinary Optimization*, Vol. 46, No. 2, 2012, pp. 273–284. doi:10.1007/s00158-012-0763-y.

[8] Zaefferer, M., and Horn, D., "A First Analysis of Kernels for Kriging-Based Optimization in Hierarchical Search Spaces," *Parallel Problem Solving from Nature, PPSN XI*, Vol. 1, edited by R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, Springer Berlin Heidelberg, Berlin, Heidelberg, 2018, pp. 399–410. doi:10.1007/978-3-319-99259-4_32.

[9] Deb, K., "Multi-Objective Optimization Using Evolutionary Algorithms," , 2001. doi:10.1109/TEVC.2002.804322.

[10] Gray, J., "MDO Problem Suite v3: We have the technology, we can rebuilt it!" Oral Presentation, Jun. 2020.

[11] Epstein, A. H., "Aeropropulsion for Commercial Aviation in the Twenty-First Century and Research Directions Needed," *AIAA Journal*, Vol. 52, No. 5, 2014, pp. 901–911. doi:10.2514/1.j052713.

[12] Oates, G., "The Aerothermodynamics of Aircraft Gas Turbine Engines," techreport ADA059784, Washington University Seattle, Jul. 1978.

[13] Aloyo, K. C., Perullo, C., and Mavris, D. N., "An Assessment of Ultra High Bypass Engine Architecture and Installation Considerations," *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, American Institute of Aeronautics and Astronautics, 2014. doi:10.2514/6.2014-3685.

[14] Guynn, M. D., Berton, J. J., Tong, M. J., and Haller, W. J., "Advanced Single-Aisle Transport Propulsion Design Options Revisited," *2013 Aviation Technology, Integration, and Operations Conference*, American Institute of Aeronautics and Astronautics, 2013. doi:10.2514/6.2013-4330.

[15] Reitenbach, S., Krumme, A., Behrendt, T., Schnös, M., Schmidt, T., Hönig, S., Mischke, R., and Mörland, E., "Design and Application of a Multidisciplinary Predesign Process for Novel Engine Concepts," *Journal of Engineering for Gas Turbines and Power*, Vol. 141, No. 1, 2018. doi:10.1115/1.4040750.

[16] Petit, O., Xisto, C., Zhao, X., and Grönstedt, T., "An Outlook for Radical Aero Engine Intercooler Concepts," *Volume 3: Coal, Biomass and Alternative Fuels; Cycle Innovations; Electric Power; Industrial and Cogeneration; Organic Rankine Cycle Power Systems*, American Society of Mechanical Engineers, 2016. doi:10.1115/gt2016-57920.

[17] Zhao, X., "Aero Engine Intercooling - Conceptual Design and Experimental Validation of an Aero Engine Intercooler," phdthesis, Chalmers University of Technology, Göthenborg, Sweden, 2016.

[18] Schweiger, F., "Dual cycle turbofan engine," *23rd Joint Propulsion Conference*, American Institute of Aeronautics and Astronautics, 1987. doi:10.2514/6.1987-2102.

[19] Díez, N. G., Rao, A. G., and van Buijtenen, J., "Conceptual Study of Counter-Rotating Turbofan Engines," *Volume 1: Aircraft Engine: Ceramics; Coal, Biomass and Alternative Fuels: Education; Electric Power; Manufacturing Materials and Metallurgy*, ASMEDC, 2010. doi:10.1115/gt2010-22770.

[20] Michel, U., "The Benefits of Variable Area Fan Nozzles on Turbofan Engines," *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics, 2011. doi:10.2514/6.2011-226.

[21] McKay, B., and Barlow, A., "The UltraFan Engine and Aircraft Based Thrust Reversing," *48th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, American Institute of Aeronautics and Astronautics, 2012. doi:10.2514/6.2012-3919.

[22] Hecken, T., Zhao, X., Iwanizki, M., Arzberger, M. J., Silberhorn, D., Plohr, M., Kyprianidis, K., Sahoo, S., Sumsurooah, S., Valente, G., Sielemann, M., Coïc, C., Bardenhagen, A., Scheunemann, A., and Jacobs, C., "Conceptual Design Studies of "Boosted Turbofan" Configuration for short range," *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, 2020. doi:10.2514/6.2020-0506.

[23] Lord, W. K., Suciu, G., Chandler, J., and Hasel, K., "Engine Architecture for High Efficiency at Small Core Size," *53rd AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, 2015. doi:10.2514/6.2015-0071.

[24] Hendricks, E. S., and Gray, J. S., "pyCycle: A Tool for Efficient Optimization of Gas Turbine Engine Cycles," *Aerospace*, Vol. 6, No. 8, 2019, p. 87. doi:10.3390/aerospace6080087.

[25] Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., and Naylor, B. A., "OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104. doi:10.1007/s00158-019-02211-z.

[26] Onat, E., and Klees, G., "A Method to Estimate Weight and Dimensions of Large and Small Gas Turbine Engines," techreport NASA-CR-159481, NASA, Jan. 1979.

[27] Torenbeek, E., *Synthesis of Subsonic Airplane Design*, Springer Netherlands, 1982.

[28] Bussemaker, J., and De Smedt, T., "jbussemaker/OpenTurbofanArchitecting: Aircraft Jet Engine Architecting Benchmark Problem," , 2021. doi:10.5281/ZENODO.5011359.

[29] Proesmans, P.-J., "Preliminary Propulsion System Design and Integration for a Box-Wing Aircraft Configuration: A Knowledge Based Engineering Approach," mathesis, Delft University of Technology, Aerospace Engineering, Dec. 2019.

[30] Greitzer, E., and Slater, H., *Design Methodologies for Aerodynamics, Structures, Weight, and Thermodynamic Cycles*, MIT, 2010.

[31] Waters, M. H., and Schairer, E. T., *Analysis of Turbofan Propulsion System Weight and Dimensions*, NASA, 1977.

[32] Torenbeek, E., and Berenschot, G., *De Berekening van het Omspoeld Gondeloppervlak van Enkel- en Dubbelstroom Straalmotoren voor Civiele Vliegtuigen*, Technische Hogeschool Delft, 1983.

[33] Dallara, E. S., "Aircraft Design for Reduced Climate Impact," , 2011. PhD.

[34] Stone, J. R., *Interim Prediction Method for Jet Noise*, NASA, 1974.

[35] GasTurb GmbH, *GasTurb 13: Design and Off-Design Performance of Gas Turbines*, 2018.

[36] Blank, J., and Deb, K., "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, Vol. 8, 2020, pp. 89497–89509. doi:10.1109/access.2020.2990567.

[37] COBRA Project Consortium, "Final Report Summary - COBRA (Innovative counter rotating fan system for high bypass ratio aircraft engine)," , 2018. URL `https://cordis.europa.eu/project/id/605379/reporting`, accessed June 2021.

[38] Yin, F., and Rao, A. G., "Performance analysis of an aero engine with inter-stage turbine burner," *The Aeronautical Journal*, Vol. 121, 2017, p. 1605–1626. doi:10.1017/aer.2017.93.

[39] *The Jet Engine*, 5th ed., Rolls-Royce, 1986.

[40] EASA, "For Engine LEAP-1A & LEAP-1C series engines," , 2018. URL https://web.archive.org/web/20181013014334/https://www.easa.europa.eu/sites/default/files/dfu/EASA%20E110%20TCDS%20Issue%207%20LEAP-1A-1C.pdf, accessed June 2021.

[41] Whitney, P. &., "F100-PW-229 Engine," , 2021. URL https://prattwhitney.com/products-and-services/products/military-engines/F100-PW-229-ENGINE, accessed June 2021.

[42] Scholz, D., Sereshine, R., Staack, I., and Lawson, C., "Fuel Consumption due to Shaft Power Off-takes from the Engine," , 2013. URL https://www.fzt.haw-hamburg.de/pers/Scholz/Off-Takes/Off-Takes_PRE_AST_13-04-23.pdf, accessed June 2021.

[43] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182–197. doi:10.1109/4235.996017.

[44] Müller, J., and Day, M., "Surrogate Optimization of Computationally Expensive Black-Box Problems with Hidden Constraints," *INFORMS Journal on Computing*, Vol. 31, No. 4, 2019, pp. 689–702. doi:10.1287/ijoc.2018.0864.

[45] *Aeronautical Technologies for the Twenty-First Century*, National Academy Press, 1992.