Chapter 7

Multidisciplinary Design Optimization

7.1 Introduction

Multidisciplinary design optimization (MDO) is a field of engineering that focuses on use of numerical optimization to perform the design of systems that involve a number of disciplines or subsystems. The main motivation for using MDO is that the best design of a multidisciplinary system can only be found when the interactions between the system's disciplines are fully considered. Considering these interactions in the design process cannot be done in an arbitrary way and requires a sound mathematical formulation. By solving the MDO problem early in the design process and taking advantage of advanced computational analysis tools, designers can simultaneously improve the design and reduce the time and cost of the design cycle.

The origins of MDO can be traced back to Schmit [135, 136, 137] and Haftka [59, 60, 62], who extended their experience in structural optimization to include other disciplines. One of the first applications of MDO was aircraft wing design, where aerodynamics, structures, and controls are three strongly coupled disciplines [54, 55, 103, 104]. Since then, the application of MDO has been extended to complete aircraft [92, 106] and a wide range of other engineering systems, such as bridges [11], buildings [35, 50], railway cars [45, 68], microscopes [128], automobiles [89, 114], ships [74, 127], rotorcraft [48, 52], and spacecraft [23, 30].

One of the most important considerations when implementing MDO is how to organize the disciplinary analysis models, approximation models (if any), and optimization software in concert with the problem formulation so that an optimal design is achieved. Such a combination of problem formulation and organizational strategy is referred to as an MDO *architecture*. The MDO architecture defines both how the different models are coupled and how the overall optimization problem is solved. The architecture can be either *monolithic* or *distributed*. In monolithic approaches, a single optimization problem is solved. In a distributed approach, the same single problem is partitioned into multiple subproblems containing small subsets of the variables and constraints.

While many different architectures can be used to solve a given optimal design problem — and just as many algorithms may be used to solve a given optimization problem — choosing the most appropriate architecture for the problem can significantly reduce the solution time. These time savings come from the selected methods for solving each discipline, the coupling scheme used in the architecture, and the degree to which operations are carried out in parallel. The latter consideration becomes especially important as the design becomes more detailed and the number of variables and/or constraints increases.

Note that in the MDO literature, there are several terms used to describe what we mean by "architecture": "method"[1, 92, 132, 150, 163], "methodology"[81, 115, 118], "problem formulation"[6,

7, 38], "strategy"[63, 166], "procedure"[84, 143] and "algorithm"[44, 139, 141, 149] have all been used. Note that some authors use a variety of terms and occasionally use them interchangeably in the same paper. Our preference for the term "architecture" [25, 33, 91, 142] comes from the fact that the relationship between problem formulation and solution algorithm is not one-to-one. For example, replacing a particular disciplinary simulation with a surrogate model or reordering the disciplinary simulations do not affect the problem formulation but strongly affect the solution algorithm.

There have been a number of surveys on MDO over the last two decades. Haftka et al. [58] were one of the first to review the MDO methods known at the time. Cramer et al. [38] formalized the monolithic architectures and detailed the required sensitivity analysis methods. Balling and Sobieski [13] identified a number of possible monolithic approaches and estimated their computational cost. In the collection of articles entitled "Multidisciplinary Design Optimization: State of the Art" and edited by Alexandrov and Hussaini [3], Kroo [91] provided a comprehensive overview of MDO, including a description of both monolithic and distributed architectures. In the same volume, Alexandrov [4] discussed the convergent properties of certain partitioning strategies, and Balling [12] focused on partitioning as a way to provide disciplinary autonomy. In the same year, Sobieski and Haftka [144] published an exhaustive survey of the MDO literature up to that time.

Since this last series of surveys, MDO has continued to be an active field of research. New architectures for MDO have been developed and various successful applications of MDO have taken place in industry [1]. A recent paper by Tosserams et al. [159] identified numerous architectures developed in the last decade that were not covered by the previous surveys. However, there is currently no comprehensive description of all existing architectures that compares the features, merits, and performance of each architecture.

The purpose of this chapter is to survey the available MDO architectures and present them in a unified notation to facilitate understanding and comparison. Furthermore, we propose the use of a new standard diagram to visualize the algorithm of a given MDO architecture, how its components are organized, and its data flow. We pay particular attention to the newer MDO architectures that have yet to gain widespread use. For each architecture, we discuss its features and expected performance. We also present a new classification of MDO architectures and show how they relate mathematically. This classification is especially novel as it is able to draw similarities between architectures that were developed independently.

This chapter is organized as follows. In §7.2 we present the unified notation and diagrams for describing MDO methods. In §7.3 we define the general MDO problem and describe the four basic monolithic architectures and their derivation. In §7.4 we focus on the distributed architectures, and include a discussion on the motivation for using such methods and their mathematical derivation. We also describe a new classification of distributed architectures by drawing parallels to the monolithic architectures and then explain the distributed MDO architectures in detail, discussing their capabilities. Finally, in §7.5 we survey some of the benchmarking studies that have been performed to help decide which architectures are most efficient for certain classes of design problems.

7.2 Unified Description of MDO Architectures

7.2.1 Terminology and Mathematical Notation

Before introducing the mathematical definition of the MDO problem or any architectures, we introduce the notation that we use throughout this chapter. This notation is useful to compare the various problem formulations within the architectures and in identifying how similar features of the general MDO problem are handled in each case. The notation is listed in Table 7.1. This is not a

Table 7.1: Mathematical notation for MDO problem formulations

Symbol	Definition
x	Vector of design variables
y^t	Vector of coupling variable targets (inputs to a discipline analysis)
y	Vector of coupling variable responses (outputs from a discipline analysis)
$\bar{y}$	Vector of state variables (variables used inside only one discipline analysis)
f	Objective function
c	Vector of design constraints
c^c	Vector of consistency constraints
$\mathcal{R}$	Governing equations of a discipline analysis in residual form
N	Number of disciplines
$n_{()}$	Length of given variable vector
$m_{()}$	Length of given constraint vector
$()_0$	Functions or variables that are shared by more than one discipline
$()_i$	Functions or variables that apply only to discipline i
$()^*$	Functions or variables at their optimal value
$\tilde{()}$	Approximation of a given function or vector of functions

comprehensive list; additional notation specific to particular architectures is introduced when the respective architectures are described. We also take this opportunity to clarify many of the terms we use that are specific to the field of MDO.

A *design variable* is a variable in the MDO problem that is always under the explicit control of an optimizer. In traditional engineering design, values of these variables are selected explicitly by the designer or design team. Design variables may pertain only to a single discipline, i.e., *local*, or may be *shared* by multiple disciplines. We denote the vector of design variables local to discipline i by x_i and shared variables by x_0. The full vector of design variables is given by $x = \left[x_0^T, x_1^T, \ldots, x_N^T\right]^T$. The subscripts for local and shared data are also used in describing objectives and constraints.

A *discipline analysis* is a simulation that models the behavior of one aspect of a multidisciplinary system. Running a discipline analysis consists in solving a system of equations — such as the Navier–Stokes equations in fluid mechanics, or the static equilibrium equations in structural mechanics — which compute a set of discipline responses, known as *state variables*. State variables may or may not be controlled by the optimization, depending on the formulation employed. We denote the vector of state variables computed within discipline i by $\bar{y}_i$. We denote the associated set of disciplinary equations in residual form by $\mathcal{R}_i$, so that the expression $\mathcal{R}_i = 0$ represents the solution of these equations with respect to $\bar{y}_i$.

In a multidisciplinary system, most disciplines are required to exchange *coupling variables* to model the interactions of the whole system. Often, the number of variables exchanged is much smaller than the total number of state variables computed in a particular discipline. For example, in aircraft design, state information about the entire flow field resulting from the aerodynamics analysis is not required by the structural analyses. Instead, only the aerodynamic loads on the aircraft surface are passed. The coupling variables supplied by a given discipline i are denoted by y_i. Another common term for y_i is *response variables*, since they describe the response of the analysis to a design decision. In general, a transformation is required to compute y_i from $\bar{y}_i$ for each discipline [38]. Similarly, a transformation may be needed to convert input coupling variables into a usable format within each discipline [38]. In this work, the mappings between y_i and $\bar{y}_i$ are

lumped into the analysis equations $\mathcal{R}_i$. This simplifies our notation with no loss of generality.

In many formulations, copies of the coupling variables must be made to allow discipline analyses or optimizations to run independently and in parallel. These copies are known as *target* variables, which we denote by a superscript t. For example, the copy of the response variables produced by discipline i is denoted y_i^t. These variables are used as the input to disciplines that are coupled to discipline i through y_i. In order to preserve consistency between the coupling variable inputs and outputs at the optimal solution, we define a set of *consistency constraints*, $c_i^c = y_i^t - y_i$, which we add to the problem formulation.

7.2.2 Architecture Diagrams — The Extended Design Structure Matrix

While rewriting problem formulations for each architecture using a common notation is a straightforward task, describing the sequence of operations in the implementation in a convenient way presents a significant challenge. Some authors just present the problem formulation and let the readers work out the implementation by themselves. This is acceptable for monolithic architectures. For some of the distributed architectures, however, the implementation is not obvious. Other authors use an algorithm or flowchart as an aid, but these are often inadequate for describing the data flow between the different software components. Furthermore, more complex strategies with multiple loops and parallel processes are difficult to describe compactly using this technique. The lack of a standard convenient graphical representation to describe the solution strategy in MDO architectures is another impediment to understanding and comparing their relative merits.

To enhance our discussion, each of the architectures is presented with a new diagram known as an Extended Design Structure Matrix, or XDSM [93]. As the name suggests, the XDSM was based on the Design Structure Matrix [28, 151], a common diagram in systems engineering that is used to visualize the interconnections among components of a complex system. The XDSM was developed to simultaneously communicate data dependency and process flow between computational components of the architecture on a single diagram. We present only a brief overview of the XDSM in this work. Further details of the diagram syntax and interpretation and other applications of the XDSM are presented by Lambe and Martins [93].

We present the XDSM using two simple examples. Figure 7.1, the first example, shows a Gauss–Seidel multidisciplinary analysis (MDA) procedure for three disciplines, which is described in Algorithm 14. The components consist of the discipline analyses themselves and a special component, known as a *driver*, which controls the iteration. The interfaces between these components consist of the data that is exchanged. Following the DSM rules, the components are placed on the main diagonal of a matrix while the interfaces are placed in the off-diagonal locations such that inputs to a component are placed in the same column and outputs are placed in the same row. External inputs and outputs may also be defined and are placed on the outer edges of the diagram. In the case of Figure 7.1, external input consists of the design variables and an initial guess of the system coupling variables. Each discipline analysis computes its own set of coupling variables which is passed to other discipline analyses or back to the driver. At the end of the MDA process, each discipline returns the final set of coupling variables computed. Thick gray lines are used to show the data flow between components.

A numbering system is used to show the order in which the components are executed. The algorithm starts at component zero and proceeds in numerical order. Loops are denoted using the notation $j \rightarrow k$ for $k < j$ so that the algorithm must return to step k until a looping condition is satisfied before proceeding. The data nodes are also labeled with numbers to denote the time at which the input data is retrieved. As an added visualization aid, consecutive components in the algorithm are connected by a thin black line. Thus, following the procedure in Figure 7.1 yields

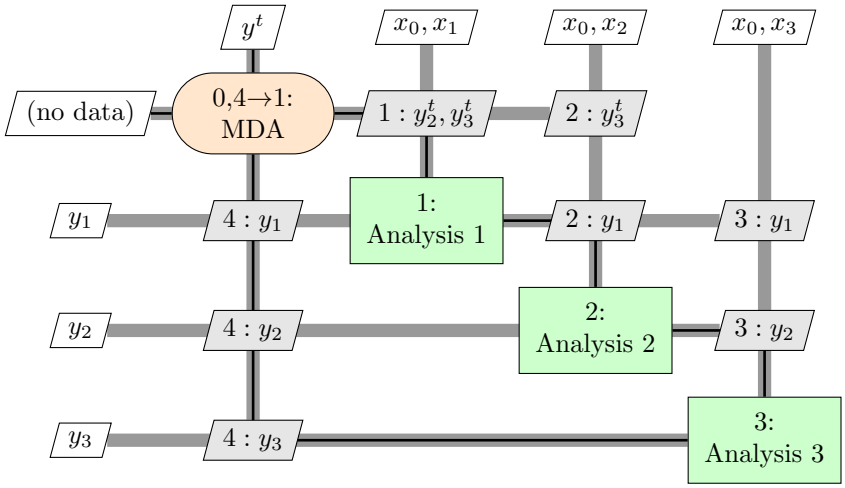

Figure 7.1: A block Gauss–Seidel multidisciplinary analysis (MDA) process to solve a three-discipline coupled system.

Algorithm 14.

Algorithm 14 Block Gauss–Seidel multidisciplinary analysis algorithm

Input: Design variables x
Output: Coupling variables, y
0: Initiate MDA iteration loop
repeat
1: Evaluate Analysis 1 and update y_1
2: Evaluate Analysis 2 and update y_2
3: Evaluate Analysis 3 and update y_3
until $4 \rightarrow 1$: MDA has converged

The second example, illustrated in Figure 7.2, is the solution process for an optimization problem using gradient-based optimization. The problem has a single objective and a vector of constraints. Figure 7.2 shows separate components to compute the objective, constraints, and their gradients and a driver to control the iteration. Notice that in this example, multiple components are evaluated at step one of the algorithm. This numbering denotes parallel execution. In some cases, it may be advisable to lump components together to reflect underlying problem structures, such as lumping together the objective and constraint components. In the following sections, we have done just that in the architecture diagrams to simplify presentation. We will also make note of other simplifications as we proceed.

7.3 Monolithic Architectures

If we ignore the disciplinary boundaries, an MDO problem is nothing more than a standard constrained nonlinear programming problem: it involves solving for the values of the design variables that maximize or minimize a particular design objective function, subject to design constraints.

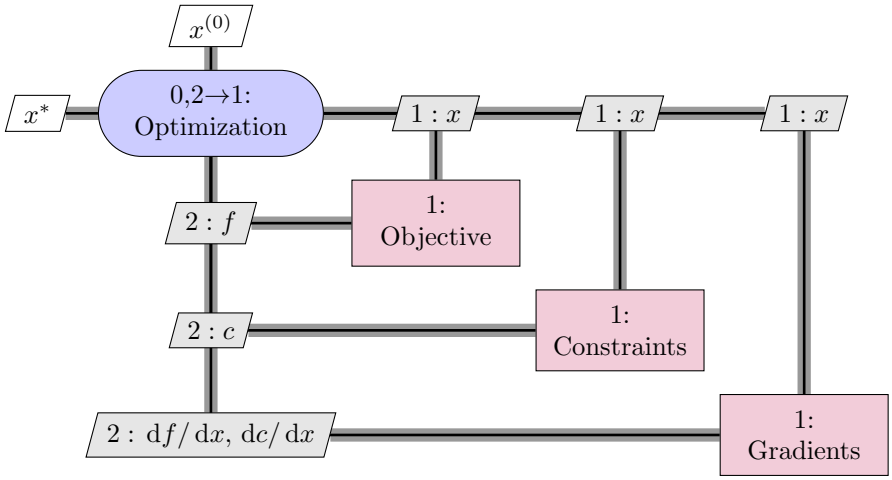

Figure 7.2: A gradient-based optimization procedure.

The choice of design objectives, design constraints, and even what variables to change in a given system is strictly up to the designer. The behavior of each component, or discipline, within the system is modeled using a discipline analysis. Each discipline analysis is usually available in the form of a computer program and can range in complexity from empirical curve-fit data to a highly detailed physics-based simulation.

One of the major challenges of MDO is how to address the coupling of the system under consideration. Like the disciplines they model, the discipline analyses themselves are mutually interdependent. A discipline analysis requires outputs of other analyses as input to resolve themselves correctly. Furthermore, the objective and constraint functions, in general, depend on both the design variables and analysis outputs from multiple disciplines. While this interdependence is sometimes ignored in practice through the use of single discipline optimizations occurring in parallel or in sequence, taking the interdependence into account leads to a more accurate representation of the behavior of the whole system. MDO architectures provide a consistent, formal setting for managing this interdependence in the design process [91].

The architectures presented in this section are referred to as *monolithic* architectures. Each architecture solves the MDO problem by casting it as single optimization problem. The differences between architectures lie in the strategies used to achieve multidisciplinary feasibility of the optimal design. Architectures that decompose the optimization problem into smaller problems, i.e., *distributed* architectures, are presented in Section 7.4.

7.3.1 The All-at-Once (AAO) Problem Statement

Before discussing specific architectures, we show the most fundamental optimization problem from which all other problem statements are derived. We can describe the MDO problem in its most

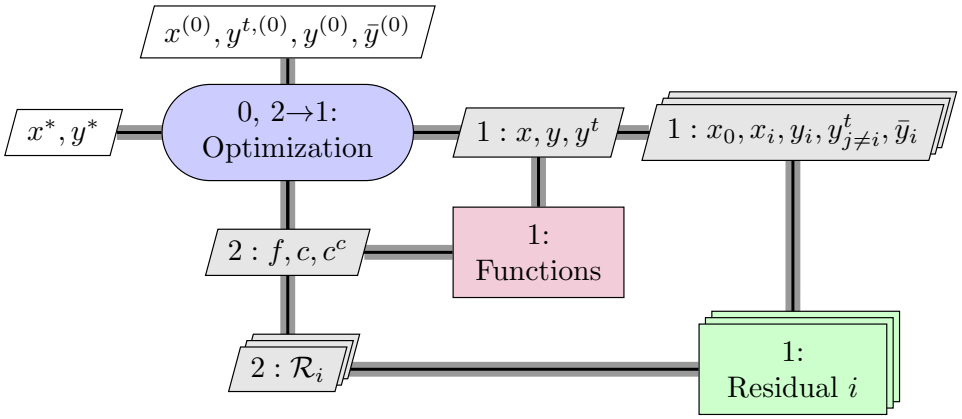

Figure 7.3: XDSM for solving the AAO problem.

general form as

$$
\begin{aligned}
\text{minimize} \quad & f_0(x,y) + \sum_{i=1}^{N} f_i(x_0, x_i, y_i) \\
\text{with respect to} \quad & x, y^t, y, \bar{y} \\
\text{subject to} \quad & c_0(x,y) \geq 0 \\
& c_i(x_0, x_i, y_i) \geq 0 && \text{for } i = 1, \ldots, N \\
& c^c_i = y^t_i - y_i = 0 && \text{for } i = 1, \ldots, N \\
& \mathcal{R}_i\left(x_0, x_i, y^t_{j\neq i}, \bar{y}_i, y_i\right) = 0 && \text{for } i = 1, \ldots, N.
\end{aligned}
\tag{7.1}
$$

For all design and state variable vectors, we use the notation $x = [x_0^T, x_1^T, \ldots, x_N^T]^T$ to concatenate the disciplinary variable groups. In future problem statements, we omit the local objective functions f_i except when necessary to highlight certain architectural features. Problem (7.1) is known as the "all-at-once" (AAO) problem. Figure 7.3 shows the XDSM for solving this problem. To keep the diagrams compact, we adopt the convention that any block referring to discipline i represents a repeated pattern for every discipline. Thus, in Figure 7.3 a residual block exists for every discipline in the problem and each block can be executed in parallel. As an added visual cue in the XDSM, the "Residual i" component is displayed as a stack of similar components.

There is a conflict with the established literature when it comes to the labeling of Problem (7.1). What most authors refer to as AAO, following the lead of Cramer et al. [38], others label as the simultaneous analysis and design (SAND) [13, 61] problem. Our AAO problem is most like what Cramer et al. refer to as simply "the most general formulation" [38]. Herein, we classify the formulation (7.1) as the AAO problem because it includes all design, state, and input and output coupling variables in the problem, so the optimizer is responsible for all variables at once. The SAND architecture uses a different problem statement and is presented in Section 7.3.2.

The AAO problem is never solved in practice because the consistency constraints, which are linear in this formulation, can be eliminated quite easily. Eliminating these constraints reduces the problem size without compromising the performance of the optimization algorithm. As we will see, eliminating the consistency constraints from Problem (7.1) results in the problem solved by the SAND architecture. However, we have presented the AAO problem first because it functions as a

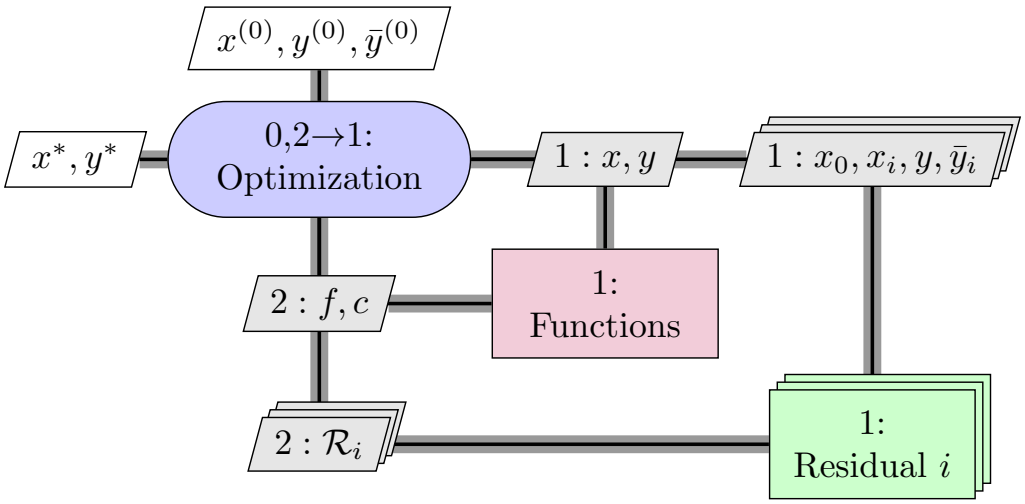

Figure 7.4: Diagram for the SAND architecture.

common starting point for deriving both the SAND problem and the Individual Discipline Feasible (IDF) problem and, subsequently, all other correctly-formulated MDO problems.

Depending on which equality constraint groups are eliminated from Problem (7.1), we can derive the other three monolithic architectures: Multidisciplinary Feasible (MDF), Individual Discipline Feasible (IDF), and Simultaneous Analysis and Design (SAND). All three have been known in the literature for a long time [13, 38, 61, 138]. In the next three subsections, we describe how each architecture is derived and the relative advantages and disadvantages of each. We emphasize that in all cases, in spite of the elements added or removed by each architecture, *we are always solving the same MDO problem.*

7.3.2 Simultaneous Analysis and Design (SAND)

The most obvious simplification of Problem (7.1) is to eliminate the consistency constraints, $c_i^c = y_i^t - y_i = 0$, by introducing a single copy of the coupling variables to replace the separate target and response copies. This simplification yields the SAND architecture [61], which solves the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\right) \\ \text{with respect to} \quad & x, y, \bar{y} \\ \text{subject to} \quad & c_0\left(x, y\right) \geq 0 \\ & c_i\left(x_0, x_i, y_i\right) \geq 0 \qquad \text{for } i = 1, \ldots, N \\ & \mathcal{R}_i\left(x_0, x_i, y, \bar{y}_i\right) = 0 \quad \text{for } i = 1, \ldots, N. \end{aligned} \tag{7.2}$$

The XDSM for SAND is shown in Figure 7.4. Cramer et al. [38] refer to this architecture as "All-at-Once". However, we use the name SAND to reflect the consistent set of analysis and design variables chosen by the optimizer. The optimizer, therefore, can simultaneously analyze and design the system.

Several features of the SAND architecture are noteworthy. Because we do not need to solve any discipline analysis explicitly or exactly, the optimization problem can potentially be solved

very quickly by letting the optimizer explore regions that are infeasible with respect to the analysis constraints. The SAND methodology is not restricted to multidisciplinary systems and can be used in single discipline optimization as well. In that case, we only need to define a single group of design constraints. If the disciplinary residual equations are simply discretized partial differential equations, the SAND problem is just a PDE-constrained optimization problem like many others in the literature. (See Biegler et al. [19] for an overview of this field.)

Two major disadvantages are still present in the SAND architecture. First, the problem formulation still requires all state variables and discipline analysis equations, meaning that large problem size and potential premature termination of the optimizer at an infeasible design can be issues in practice. Second, and more importantly, the fact that the discipline analyses equations are treated as explicit constraints means that the residual values — and possibly their derivatives — need to be available to the optimizer. In engineering design, many discipline analysis codes operate in a "black-box" fashion, directly computing the coupling variables while hiding the discipline analyses residuals and state variables in the process. Even if the source code for the discipline analysis can be modified to return residuals, the cost and effort required often eliminates this option from consideration. Therefore, most practical MDO problems require an architecture that can take advantage of existing discipline analysis codes. The following two monolithic architectures address this concern.

7.3.3 Individual Discipline Feasible (IDF)

By eliminating the disciplinary analysis constraints $\mathcal{R}_i(x_0, x_i, y_i, y^t_{j\neq i}, \bar{y}_i) = 0$ from Problem (7.1), we obtain the IDF architecture [38]. As commonly noted in the literature, this type of elimination is achieved by applying the Implicit Function Theorem to the $\mathcal{R}_i$ constraints so that $\bar{y}_i$ and y_i become functions of design variables and coupling targets. The IDF architecture is also known as distributed analysis optimization [6] and optimizer-based decomposition [91]. The optimization problem for the IDF architecture is

$$\begin{aligned}
\text{minimize} \quad & f_0\left(x, y\left(x, y^t\right)\right) \\
\text{with respect to} \quad & x, y^t \\
\text{subject to} \quad & c_0\left(x, y\left(x, y^t\right)\right) \geq 0 \\
& c_i\left(x_0, x_i, y_i\left(x_0, x_i, y^t_{j\neq i}\right)\right) \geq 0 \quad \text{for } i = 1, \ldots, N \\
& c^c_i = y^t_i - y_i\left(x_0, x_i, y^t_{j\neq i}\right) = 0 \quad \text{for } i = 1, \ldots, N.
\end{aligned} \tag{7.3}$$

The most important consequence of this reformulation is the removal of all state variables and discipline analysis equations from the problem statement. All coupling variables are now implicit functions of design variables and coupling variable targets as a result of solving the discipline analyses equations exactly each time.

The XDSM for IDF is shown in Figure 7.5. This architecture enables the discipline analyses to be performed in parallel, since the coupling between the disciplines is resolved by the target coupling variables, y^t, and consistency constraints, c^c. Within the optimization iteration, specialized software for solving the discipline analyses can now be used to return coupling variable values to the objective and constraint function calculations. The net effect is that the IDF problem is both substantially smaller than the SAND problem and requires minimal modification to existing discipline analyses. In the field of PDE-constrained optimization, the IDF architecture is exactly analogous to a reduced-space method [19].

In spite of the reduced problem size when compared to SAND, the size of the IDF problem can still be an issue. If the number of coupling variables is large, the size of the resulting optimization

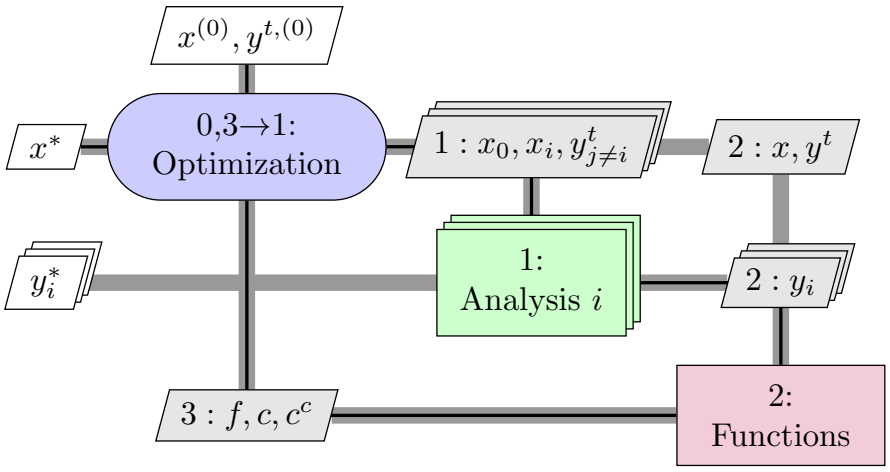

Figure 7.5: Diagram of the IDF architecture.

problem might still be too large to solve efficiently. The large problem size can be mitigated to some extent by careful selection of the partitioning strategy or aggregation of the coupling variables to reduce information transfer between disciplines.

A bigger issue in IDF concerns gradient computation. If gradient-based optimization software is used — which is likely because it is generally more efficient for large problems — evaluating the objective and constraint function gradients becomes a costly part of the optimization procedure. This is because the gradients themselves must be discipline-feasible, i.e., the changes in design variables cannot cause the output coupling variables to violate the discipline analysis equations to first order. The errors caused by inaccurate gradient values can severely impact the performance of gradient-based optimization.

In practice, gradients are often calculated using some type of finite-differencing procedure, where the discipline analysis is evaluated for each design variable. While this approach preserves disciplinary feasibility, it is costly and unreliable. If the discipline analysis code allows for the use of complex numbers, the complex-step method [109] is an alternative approach which gives machine-precision derivative estimates. If the analysis codes require a particularly long time to evaluate, the use of automatic differentiation or analytic derivative calculations (i.e. direct or adjoint methods) can be used to avoid multiple discipline analysis evaluations [105]. While the development time for these methods can be long, the reward is accurate derivative estimates and massive reductions in computational cost, especially for design optimization based on high-fidelity models.

7.3.4 Multidisciplinary Feasible (MDF)

If both analysis and consistency constraints are removed from Problem (7.1), we obtain the MDF architecture [38]. This architecture has also been referred to in the literature as Fully Integrated

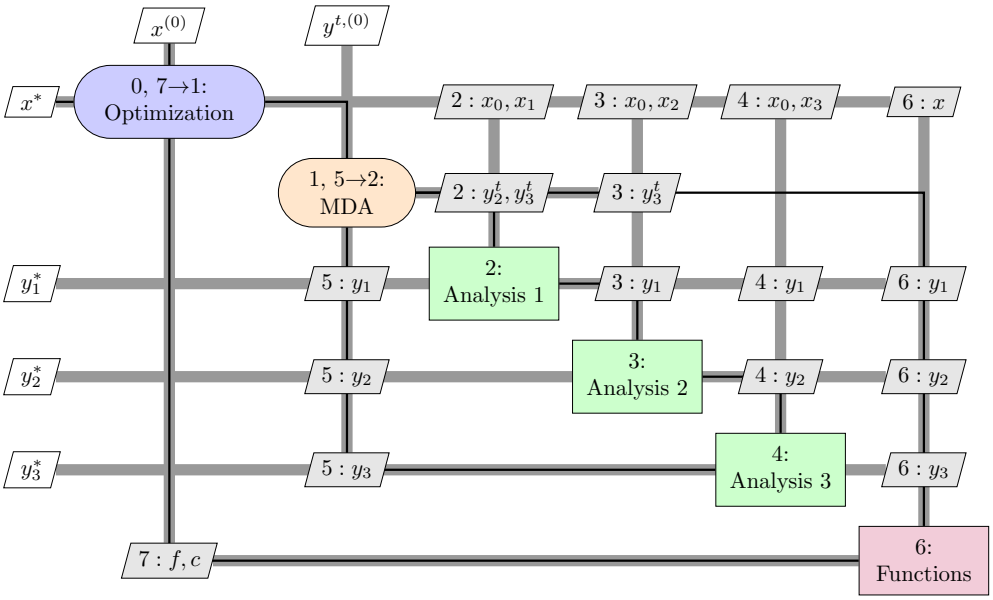

Figure 7.6: Diagram for the MDF architecture with a Gauss–Seidel multidisciplinary analysis.

Optimization [6] and Nested Analysis and Design [13]. The resulting optimization problem is

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\left(x, y\right)\right) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & c_0\left(x, y\left(x, y\right)\right) \geq 0 \\ & c_i\left(x_0, x_i, y_i\left(x_0, x_i, y_{j\neq i}\right)\right) \geq 0 \quad \text{for } i = 1, \ldots, N. \end{aligned} \tag{7.4}$$

The MDF architecture XDSM is shown in Figure 7.6. Typically, a fixed point iteration, such as the block Gauss–Seidel iteration shown in Figure 7.6, is used to converge the multidisciplinary analysis (MDA), where each discipline is solved in turn. This is usually an approach that exhibits slow convergence rates. Re-ordering the sequence of disciplines can improve the convergence rate of Gauss–Seidel [21], but even better convergence rates can be achieved through the use of Newton-based methods [75]. Note that due to the sequential nature of the Gauss–Seidel iteration, we cannot evaluate the disciplines in parallel and cannot apply our convention for compacting the XDSM. Using a different MDA method results in a different XDSM.

An obvious advantage of MDF over the other monolithic architectures is that the optimization problem is as small as it can be for a monolithic architecture, since only the design variables and design constraints are under the direct control of the optimizer. Another benefit is that MDF always returns a fully consistent system design, even if the optimization process is terminated early. This is advantageous in an engineering design context if time is limited and we are not as concerned with finding a mathematically optimal design as with finding an improved design. Note, however, that design constraint satisfaction is not guaranteed if the optimization is terminated early; that depends on whether the optimization algorithm maintains a feasible design point or not. In particular, methods of feasible directions [164] require and maintain a feasible design point while many robust sequential quadratic programming [51] and interior point methods [165] do not.

The main disadvantage of MDF is that a consistent set of coupling variables must be computed and returned to the optimizer every time the objective and constraint functions are re-evaluated.

In other words, the architecture requires a full MDA to be performed for every optimizer iteration. Instead of simply running each individual discipline analysis once per optimizer iteration, as we do in IDF, we need to run every discipline analysis multiple times until a consistent set of coupling variables is found. This task requires its own specialized iterative procedure outside of the optimization. Developing an MDA procedure can be time consuming if one is not already in place.

Gradient calculations are also much more difficult for MDF than for IDF. Just as the gradient information in IDF must be discipline-feasible, the gradient information under MDF must be feasible with respect to all disciplines. Fortunately, research in the sensitivity of coupled systems is fairly mature, and semi-analytic methods are available to drastically reduce the cost of this step by eliminating finite differencing over the full MDA [111, 146]. There is also some preliminary work towards automating the implementation of these coupled sensitivity methods [108]. The required partial derivatives can be obtained using any of the methods described in Section 7.3.3 for the individual disciplines in IDF.

7.4 Distributed Architectures

7.4.1 Motivation

Thus far, we have focused our discussion on monolithic MDO architectures: those that form and solve a single optimization problem. Many more architectures have been developed that decompose this single optimization problem into a set of smaller optimization problems, or subproblems, that have the same solution when reassembled. These are the *distributed* MDO architectures. Before reviewing and classifying the distributed architectures, we discuss the motivation of MDO researchers in developing this new class of MDO architectures.

Early in the history of optimization, the motivation for decomposition methods was to exploit the structure of the problem to reduce solution time. Many large optimization problems, such as network flow problems and resource allocation problems, exhibit such special structures [94].

To better understand decomposition, consider the following problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{with respect to} \quad & x_1, \ldots, x_N \\ \text{subject to} \quad & c_0(x_1, \ldots, x_N) \leq 0 \\ & c_1(x_1) \leq 0, \ldots, c_N(x_N) \leq 0. \end{aligned} \tag{7.5}$$

In this problem, there are no shared design variables, x_0, and the objective function is separable, i.e. it can be expressed as a sum of functions, each of which depend only on the corresponding local design variables, x_i. On the other hand, the constraints include a set of constraints, c_0, that depends on more than one set of design variables. This problem is referred to as a *complicating constraints problem* [36]; if c_0 did not exist, we could simply decompose this optimization problem into N independent problems.

Another possibility is that a problem includes shared design variables and a separable objective

function, with no complicating constraints, i.e.,

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_0, x_i) \\ \text{with respect to} \quad & x_0, x_1, \ldots, x_N \\ \text{subject to} \quad & c_1(x_0, x_1) \leq 0, \ldots, c_N(x_0, x_N) \leq 0. \end{aligned} \tag{7.6}$$

This is referred to as a problem with *complicating variables* [36]. In this case, the decomposition would be straightforward if there were no shared design variables, x_0, and we could solve N optimization problems independently and in parallel.

Specialized decomposition methods were developed to reintroduce the complicating variables or constraints into these problems with only small increases in time and cost relative to the N independent problems. Examples of these methods include Dantzig–Wolfe decomposition [39] and Benders decomposition [15] for Problems (7.5) and (7.6), respectively. However, these decomposition methods were designed to work with the simplex algorithm on linear programming problems. In the simplex algorithm, the active set changes only by one constraint at a time so decomposition is the only way to exploit the special problem structure. However, algorithms for nonlinear optimization that are based on Newton's method, such as sequential quadratic programming and interior point methods, may also use specialized matrix factorization techniques to exploit sparsity structures in the problem. While nonlinear decomposition algorithms were later developed, to the best of our knowledge, no performance comparisons have been made between these decomposition algorithms and Newton-like algorithms employing sparse matrix factorization. Intuition suggests that the latter should be faster due to the ability of Newton methods to exploit second-order problem information. Thus, while decomposition methods do exist for nonlinear problems, problem structure is not the primary motivation for their development.

The primary motivation for decomposing the MDO problem comes from the structure of the engineering design environment. Typical industrial practice involves breaking up the design of a large system and distributing aspects of that design to specific engineering groups. These groups may be geographically distributed and may only communicate infrequently. More importantly, however, these groups typically like to retain control of their own design procedures and make use of in-house expertise, rather than simply passing on discipline analysis results to a central design authority [91]. Decomposition through distributed architectures allow individual design groups to work in isolation, controlling their own sets of design variables, while periodically updating information from other groups to improve their aspect of the overall design. This approach to solving the problem conforms more closely with current industrial design practice than the approach of the monolithic architectures.

The structure of disciplinary design groups working in isolation has a profound effect on the timing of each discipline analysis evaluation. In a monolithic architecture, all discipline analysis programs are run exactly the same number of times, based on requests from the optimizer or MDA program. In the context of parallel computing, this approach can be thought of as a synchronous algorithm [18]. In instances where some analyses or optimizations are much more expensive than others, such as the case of multifidelity optimization [131, 168], the performance suffers because the processors performing the inexpensive analyses and optimizations experience long periods of inactivity while waiting to update their available information. In the language of parallel computing, the computation is said to exhibit poor load balancing. Another example of this case is aerostructural optimization, in which a nonlinear aerodynamics solver may require an order of magnitude more time to run than a linear structural solver [34]. By decomposing the optimization problem,

the processor work loads may be balanced by allowing disciplinary analyses with lower computational cost to perform more optimization on their own. Those disciplines with less demanding optimizations may also be allowed to make more progress before updating nonlocal information. In other words, the whole design process occurs not only in parallel but also asynchronously. While the asynchronous design process may result in more total computational effort, the intrinsically parallel nature of architecture allows much of the work to proceed concurrently, reducing the wall clock time of the optimization.

7.4.2 Classification

We now introduce a new approach to classifying MDO architectures. Some of the previous classifications of MDO architectures were based on observations of which constraints were available to the optimizer to control [5, 13]. Alexandrov and Lewis [5] used the term "closed" to denote when a set of constraints cannot be satisfied by explicit action of the optimizer, and "open" otherwise. For example, the MDF architecture is closed with respect to both analysis and consistency constraints, because their satisfaction is determined through the process of converging the MDA. Similarly, IDF is closed analysis but open consistency since the consistency constraints can be satisfied by the optimizer adjusting the coupling targets and design variables. Tosserams et al. [159] expanded on this classification scheme by discussing whether or not distributed architectures used open or closed local design constraints in the system subproblem. Closure of the constraints is an important consideration when selecting an architecture because most robust optimization software will permit the exploration of infeasible regions of the design space. Such exploration can result in faster solutions via fewer optimizer iterations but this must be weighed against the increased optimization problem size and the risk of terminating the optimization at an infeasible point.

The central idea in our classification is that distributed MDO architectures can be classified based on their monolithic analogues: either MDF, IDF, or SAND. This stems from the different approaches to handling the state and coupling variables in the monolithic architectures. It is similar to the previous classifications in that an equality constraint must be removed from the optimization problem — i.e., closed — for every variable removed from the problem statement. However, using a classification based on the monolithic architectures makes it much easier to see the connections between distributed architectures, even when these architectures are developed in isolation from each other. In many cases, the problem formulations in the distributed architecture can be derived directly from that of the monolithic architecture by adding certain elements to the problem, by making certain assumptions, and by applying a specific decomposition scheme. This classification can also be viewed as a framework in which we can develop new distributed architectures, since the starting point for a distributed architecture is *always* a monolithic architecture.

This classification of architectures is represented in Figure 7.7. Known relationships between the architectures are shown by arrows. Due to the large number of adaptations created for some distributed architectures, such as the introduction of surrogate models and variations to solve multiobjective problems, we have only included the "core" architectures in our diagram. Details on the available variations for each distributed architecture are presented in the relevant sections.

Note that none of the distributed architectures developed to date have been considered analogues of SAND. As discussed in Section 7.3, the desire to use independent "black-box" computer codes for the disciplinary analyses necessarily excluded consideration of the SAND problem formulation as a starting point. Nevertheless, the techniques used to derive distributed architectures from IDF and MDF may also be useful when using SAND as a foundation.

Our classification scheme does not distinguish between the different solution techniques for the distributed optimization problems. For example, we have not focused on the order in which the

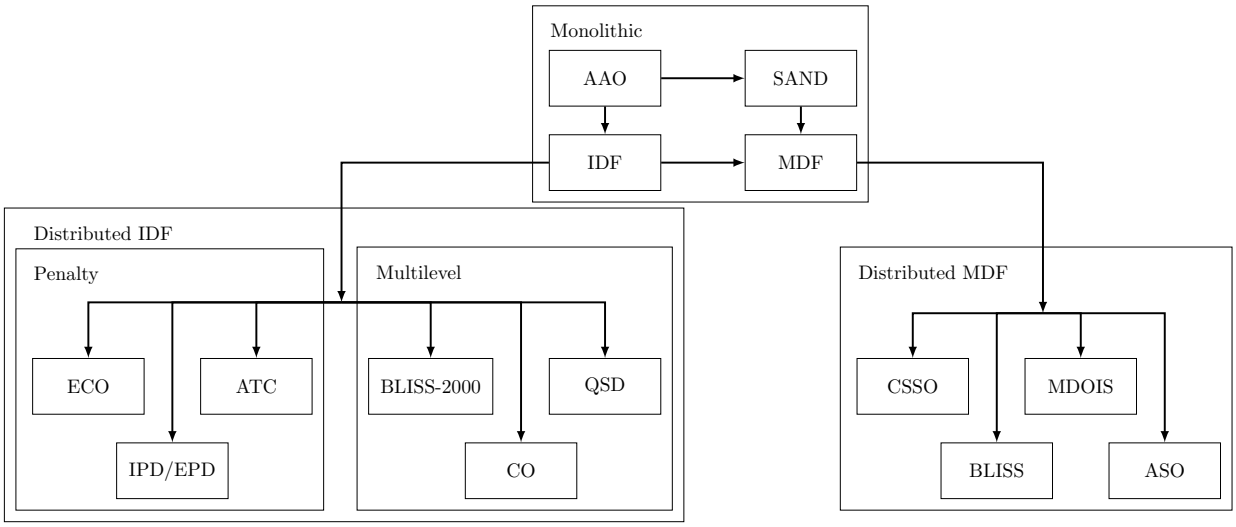

Figure 7.7: Classification of the MDO architectures.

distributed problems are solved. Coordination schemes are partially addressed in the Distributed IDF group, where we have classified the architectures as either "penalty" or "multilevel", based on whether penalty functions or a problem hierarchy is used in the coordination. This grouping follows from the work of de Wit and van Keulen [41].

One area that is not well explored in MDO is the use of hybrid architectures. By hybrid, we mean an architecture that incorporates elements of two or more other architectures in such a way that different disciplinary analyses or optimizations are treated differently. For example, a hybrid monolithic architecture could be created from MDF and IDF by resolving the coupling of some disciplines within an MDA, while the remaining coupling variables are resolved through constraints. Some ideas for hybrid architectures have been proposed by Marriage and Martins [108] and Geethaikrishnan et al. [49]. Such architectures could be especially useful in specific applications where the coupling characteristics vary widely among the disciplines. However, general rules need to be developed to say under what conditions the use of certain architectures is advantageous. As we note in Section 7.5, much work remains in this area.

In the following sections, we introduce the distributed architectures for MDO. We prefer to use the term "distributed" as opposed to "hierarchical" or "multilevel" because these architectures do not necessarily create a hierarchy of problems to solve. In some cases, it is better to think of all optimization problems as being on the same level. Furthermore, neither the systems being designed nor the design team organization need to be hierarchical in nature for these architectures to be applicable. Our focus here is to provide a unified description of these architectures and explain some advantages and disadvantages of each. Along the way, we will point out variations and applications of each architecture that can be found in the literature. We also aim to review the state-of-the-art in architectures, since the most recent detailed architecture survey in the literature dates back from more than a decade ago [143]. More recent surveys, such as that of Agte et al. [1], discuss MDO more generally without detailing the architectures themselves.

7.4.3 Concurrent Subspace Optimization (CSSO)

CSSO is one of the oldest distributed architectures for large-scale MDO problems. The original formulation [20, 145] decomposes the system problem into independent subproblems with disjoint sets of variables. Global sensitivity information is calculated at each iteration to give each subproblem a linear approximation to a multidisciplinary analysis, improving the convergence behavior. At the system level, a coordination problem is solved to recompute the "responsibility", "tradeoff", and "switch" coefficients assigned to each discipline to provide information on design variable preferences for nonlocal constraint satisfaction. Using these coefficients gives each discipline a certain degree of autonomy within the system as a whole.

Several variations of this architecture have been developed to incorporate metamodels [130, 139, 166] and higher-order information sharing among the disciplines [129]. More recently, the architecture has also been adapted to solve multiobjective problems [69, 122, 170]. Parashar and Bloebaum [123] extended a multiobjective CSSO formulation to handle robust design optimization problems. An application of the architecture to the design of high-temperature aircraft engine components is presented by Tappeta et al. [154].

The version we consider here, due to Sellar et al. [139], uses metamodel representations of each disciplinary analysis to efficiently model multidisciplinary interactions. Using our unified notation, the CSSO system subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, \tilde{y}\left(x, \tilde{y}\right)\right) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & c_0\left(x, \tilde{y}\left(x, \tilde{y}\right)\right) \geq 0 \\ & c_i\left(x_0, x_i, \tilde{y}_i\left(x_0, x_i, \tilde{y}_{j\neq i}\right)\right) \geq 0 \,\text{for}\, i = 1, \ldots, N \end{aligned} \tag{7.7}$$

and the discipline i subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y_i\left(x_i, \tilde{y}_{j\neq i}\right), \tilde{y}_{j\neq i}\right) \\ \text{with respect to} \quad & x_0, x_i \\ \text{subject to} \quad & c_0\left(x, \tilde{y}\left(x, \tilde{y}\right)\right) \geq 0 \\ & c_i\left(x_0, x_i, y_i\left(x_0, x_i, \tilde{y}_{j\neq i}\right)\right) \geq 0 \\ & c_j\left(x_0, \tilde{y}_j\left(x_0, \tilde{y}\right)\right) \geq 0 \quad \text{for}\, j = 1, \ldots, N \, j \neq i. \end{aligned} \tag{7.8}$$

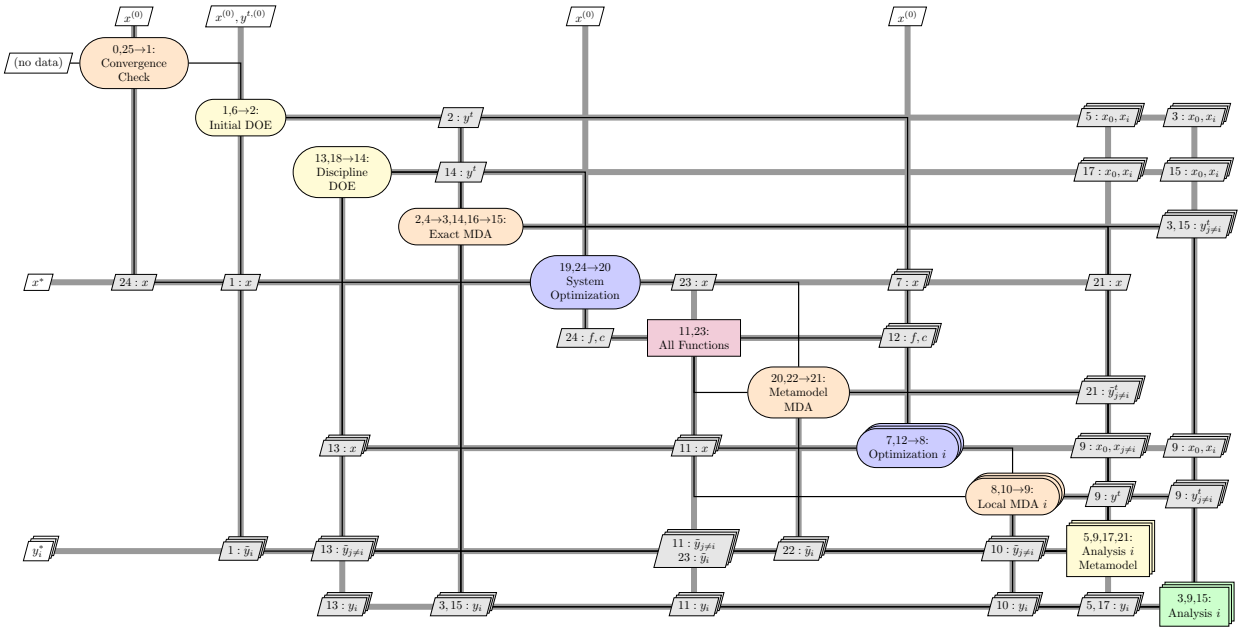

Figure 7.8: Diagram for the CSSO architecture.

The CSSO architecture is depicted in Figure 7.8 and the corresponding steps are listed in Algorithm 15. A potential pitfall of this architecture is the necessity of including all design variables in the system subproblem. For industrial-scale design problems, this may not always be possible or practical.

Algorithm 15 CSSO

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate main CSSO iteration
repeat
1: Initiate a design of experiments (DOE) to generate design points
for Each DOE point **do**
2: Initiate an MDA that uses exact disciplinary information
repeat
3: Evaluate discipline analyses
4: Update coupling variables y
until $4 \rightarrow 3$: MDA has converged
5: Update the disciplinary metamodels with the latest design
end for $6 \rightarrow 2$
7: Initiate independent disciplinary optimizations (in parallel)
for Each discipline i **do**
repeat
8: Initiate an MDA with exact coupling variables for discipline i and approximate coupling variables for the other disciplines
repeat
9: Evaluate discipline i outputs y_i, and metamodels for the other disciplines, $\tilde{y}_{j \neq i}$
until $10 \rightarrow 9$: MDA has converged
11: Compute objective f_0 and constraint functions c using current data
until $12 \rightarrow 8$: Disciplinary optimization i has converged
end for
13: Initiate a DOE that uses the subproblem solutions as sample points
for Each subproblem solution i **do**
14: Initiate an MDA that uses exact disciplinary information
repeat
15: Evaluate discipline analyses.
until $16 \rightarrow 15$ MDA has converged
17: Update the disciplinary metamodels with the newest design
end for $18 \rightarrow 14$
19: Initiate system-level optimization
repeat
20: Initiate an MDA that uses only metamodel information
repeat
21: Evaluate disciplinary metamodels
until $22 \rightarrow 21$: MDA has converged
23: Compute objective f, and constraint function values c
until $24 \rightarrow 20$: System level problem has converged
until $25 \rightarrow 1$: CSSO has converged

There have been some benchmarks comparing CSSO with other MDO architectures. Perez et al., [125] Yi et al. [167], and Tedford and Martins [155] all show CSSO requiring many more analysis calls than other architectures to converge to an optimal design. The results of de Wit and van Keulen [40] showed that CSSO was unable to reach the optimal solution of even a simple

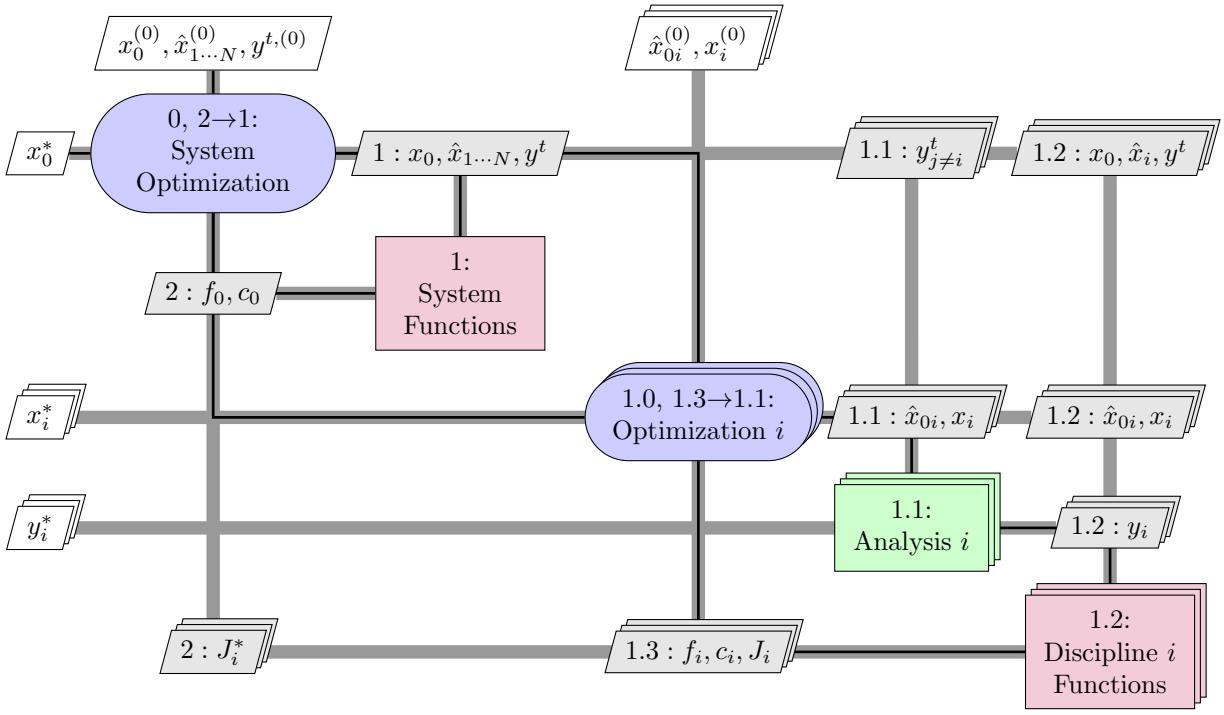

Figure 7.9: Diagram for the CO architecture.

minimum-weight two-bar truss problem. Thus, CSSO seems to be largely ineffective when compared with newer MDO architectures.

7.4.4 Collaborative Optimization (CO)

In CO, the disciplinary optimization problems are formulated to be independent of each other by using target values of the coupling and shared design variables [24, 26]. These target values are then shared with all disciplines during every iteration of the solution procedure. The complete independence of disciplinary subproblems combined with the simplicity of the data-sharing protocol makes this architecture attractive for problems with a small amount of shared data.

The XDSM for CO is shown in Figure 7.9. Braun [24] formulated two versions of the CO architecture: CO_1 and CO_2. CO_2 is the most frequently used of these two original formulations so it will be the focus of our discussion. The CO_2 system subproblem is given by:

$$\begin{aligned} \text{minimize} \quad & f_0\left(x_0, \hat{x}_1, \ldots, \hat{x}_N, y^t\right) \\ \text{with respect to} \quad & x_0, \hat{x}_1, \ldots, \hat{x}_N, y^t \\ \text{subject to} \quad & c_0\left(x_0, \hat{x}_1, \ldots, \hat{x}_N, y^t\right) \geq 0 \\ & J_i^* = ||\hat{x}_{0i} - x_0||_2^2 + ||\hat{x}_i - x_i||_2^2 + \\ & \qquad ||y_i^t - y_i\left(\hat{x}_{0i}, x_i, y_{j\neq i}^t\right)||_2^2 = 0 \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{7.9}$$

where $\hat{x}_{0i}$ are copies of the global design variables passed to discipline i and $\hat{x}_i$ are copies of the local design variables passed to the system subproblem. Note that copies of the local design variables are only made if those variables directly influence the objective. Mathematically speaking, that

means $\partial f_0/\partial x_i \neq 0$. In CO_1, the quadratic equality constraints are replaced with linear equality constraints for each target-response pair. In either case, post-optimality sensitivity analysis, i.e. computing derivatives with respect to an optimized function, is required to evaluate the derivatives of the consistency constraints J_i^*.

The discipline i subproblem in both CO_1 and CO_2 is

$$\begin{aligned} \text{minimize} \quad & J_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y^t_{j\neq i}\right)\right) \\ \text{with respect to} \quad & \hat{x}_{0i}, x_i \\ \text{subject to} \quad & c_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y^t_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{7.10}$$

Thus the system-level problem is responsible for minimizing the design objective, while the discipline level problems minimize system inconsistency. Braun [24] showed that the CO problem statement is mathematically equivalent to the IDF problem statement (7.3) and, therefore, equivalent to the original MDO problem (7.1) as well. CO is depicted in Figure 7.9 and the corresponding procedure is detailed in Algorithm 16.

Algorithm 16 Collaborative optimization

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

```
0: Initiate system optimization iteration
repeat
  1: Compute system subproblem objectives and constraints
  for Each discipline i (in parallel) do
    1.0: Initiate disciplinary subproblem optimization
    repeat
      1.1: Evaluate disciplinary analysis
      1.2: Compute disciplinary subproblem objective and constraints
      1.3: Compute new disciplinary subproblem design point and J_i
    until 1.3 → 1.1: Optimization i has converged
  end for
  2: Compute a new system subproblem design point
until 2 → 1: System optimization has converged
```

In spite of the organizational advantage of having fully separate disciplinary subproblems, CO has major weaknesses in the mathematical formulation that lead to poor performance in practice [6, 43]. In particular, the system problem in CO_1 has more equality constraints than variables, so if the system cannot be made fully consistent, the system subproblem is infeasible. This can also happen in CO_2, but it is not the most problematic issue. The most significant difficulty with CO_2 is that the constraint gradients of the system problem at an optimal solution are all zero vectors. This represents a breakdown in the constraint qualification of the Karush–Kuhn–Tucker optimality conditions, which slows down convergence for most gradient-based optimization software [6]. In the worst case, the CO_2 formulation may not converge at all. These difficulties with the original formulations of CO have inspired several researchers to develop modifications to improve the behavior of the architecture.

In a few cases, problems have been solved with CO and a gradient-free optimizer, such as a genetic algorithm [169], or a gradient-based optimizer that does not use the Lagrange multipliers in the termination condition [100] to handle the troublesome constraints. While such approaches

do avoid the obvious problems with CO, they bring other issues. Gradient-free optimizers are computationally expensive and can become the bottleneck within the CO architecture. Gradient-based optimizers that do not terminate based on Lagrange multiplier values, such as feasible direction methods, often fail in nonconvex feasible regions. As pointed out by DeMiguel [43], the CO system subproblem is set-constrained, i.e., nonconvex, because of the need to satisfy optimality in the disciplinary subproblems.

The approach taken by DeMiguel and Murray [43] to fix the problems with CO is to relax the troublesome constraints using an L_1 exact penalty function with a fixed penalty parameter value and add elastic variables to preserve the smoothness of the problem. This revised approach is called Modified Collaborative Optimization (MCO). This approach satisfies the requirement of mathematical rigor, as algorithms using the penalty function formulation are known to converge to an optimal solution under mild assumptions [47, 119]. However, the test results of Brown and Olds [27] show strange behavior in a practical design problem. In particular, they observed that for values of the penalty parameter above a threshold value, the problem could not improve the initial design point. Below a lower threshold value, the architecture showed very poor convergence. Finally, a penalty parameter could not be found which produced a final design close to those computed by other architectures. In light of these findings, the authors rejected MCO from further testing.

Another idea, proposed by Sobieski and Kroo [142], uses surrogate models, also known as metamodels, to approximate the post-optimality behavior of the disciplinary subproblems in the system subproblem. This both eliminates the post-optimality sensitivity calculation and improves the treatment of the consistency constraints. While the approach does seem to be effective for the problems they solve, to our knowledge, it has not been adopted by any other researchers to date.

The simplest and most effective known fix for the difficulties of CO involves relaxing the system subproblem equality constraints to inequalities with a relaxation tolerance, which was originally proposed by Braun et al. [26]. This approach was also successful in other test problems [107, 124], where the choice of tolerance is a small fixed number, usually 10^{-6}. The effectiveness of this approach stems from the fact that a positive inconsistency value causes the gradient of the constraint to be nonzero if the constraint is active, eliminating the constraint qualification issue. Nonzero inconsistency is not an issue in a practical design setting provided the inconsistency is small enough such that other errors in the computational model dominate at the final solution. Li et al. [97] build on this approach by adaptively choosing the tolerance during the solution procedure so that the system-level problem remains feasible at each iteration. This approach appears to work when applied to the test problems in [6], but has yet to be verified on larger test problems.

Despite the numerical issues, CO has been widely implemented on a number of MDO problems. Most of applications are in the design of aerospace systems. Examples include the design of launch vehicles [23], rocket engines [30], satellite constellations [29], flight trajectories [25, 95], flight control systems [126], preliminary design of complete aircraft [92, 106], and aircraft family design [8]. Outside aerospace engineering, CO has been applied to problems involving automobile engines [114], bridge design [11], railway cars [45], and even the design of a scanning optical microscope [128].

Adaptations of the CO architecture have also been developed for multiobjective, robust, and multifidelity MDO problems. Multiobjective formulations of CO were first described by Tappeta and Renaud [153]. McAllister et al. [113] present a multiobjective approach using linear physical programming. Available robust design formulations incorporate the decision-based models of Gu et al. [56] and McAllister and Simpson [114], the implicit uncertainty propagation method of Gu et al. [57], and the fuzzy computing models of Huang et al. [72]. Multiple model fidelities were integrated into CO for an aircraft design problem by Zadeh and Toropov [168].

The most recent version of CO — Enhanced Collaborative Optimization (ECO) — was devel-

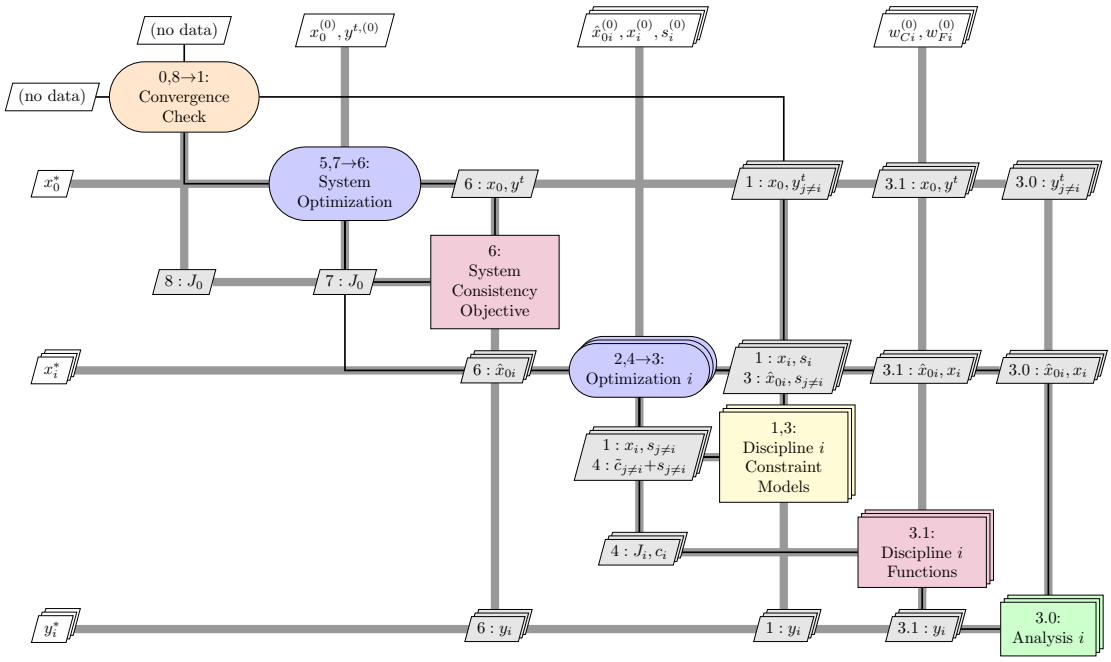

Figure 7.10: XDSM for the ECO architecture

oped by Roth and Kroo [132, 133]. Figure 7.10 shows the XDSM corresponding to this architecture. The problem formulation of ECO, while still being derived from the same basic problem as the original CO architecture, is radically different and therefore deserves additional attention. In a sense, the roles of the system and discipline optimization have been reversed in ECO when compared to CO. In ECO the system subproblem minimizes system infeasibility, while the disciplinary subproblems minimize the system objective. The system subproblem is

$$\begin{aligned} \text{minimize} \quad & J_0 = \sum_{i=1}^{N} ||\hat{x}_{0i} - x_0||_2^2 + ||y_i^t - y_i\left(x_0, x_i, y^t_{j\neq i}\right)||_2^2 \\ \text{with respect to} \quad & x_0, y^t. \end{aligned} \tag{7.11}$$

Note that this subproblem is unconstrained. Also, unlike CO, post-optimality sensitivities are not required by the system subproblem because the disciplinary responses are treated as parameters. The system subproblem chooses the shared design variables by averaging all disciplinary preferences.

The i^{th} disciplinary subproblem is

$$\begin{aligned}
\text{minimize} \quad & J_i = \tilde{f}_0\left(\hat{x}_{0i}, y_i\left(\hat{x}_{0i}, x_i, y^t_{j\neq i}\right)\right) + \\
& w_{Ci}\left(||\hat{x}_{0i} - x_0||_2^2 + ||y_i^t - y_i\left(\hat{x}_{0i}, x_i, y^t_{j\neq i}\right)||_2^2\right) + \\
& w_{Fi}\sum_{j=1, j\neq i}^{N}\sum_{k=1}^{n_s} s_{jk} \\
\text{with respect to} \quad & \hat{x}_{0i}, x_i, s_{j\neq i} \\
\text{subject to} \quad & c_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y^t_{j\neq i}\right)\right) \geq 0 \\
& \tilde{c}_{j\neq i}\left(\hat{x}_{0i}\right) - s_{j\neq i} \geq 0 \qquad j = 1, \ldots, N \\
& s_{j\neq i} \geq 0 \qquad j = 1, \ldots, N,
\end{aligned} \tag{7.12}$$

where w_{Ci} and w_{Fi} are penalty weights for the consistency and nonlocal design constraints, and s is a local set of elastic variables for the constraint models. The w_{Fi} penalty weights are chosen to be larger than the largest Lagrange multiplier, while the w_{Ci} weights are chosen to guide the optimization toward a consistent solution. Theoretically, each w_{Ci} must be driven to infinity to enforce consistency exactly. However, smaller finite values are used in practice to both provide an acceptable level of consistency and explore infeasible regions of the design space [132].

The main new idea introduced in ECO is to include linear models of nonlocal constraints, represented by $\tilde{c}_{j\neq i}$, and a quadratic model of the system objective function in each disciplinary subproblem, represented by $\tilde{f}_0$. This is meant to increase each discipline's "awareness" of their influence on other disciplines and the global objective as a whole. The construction of the constraint models deserves special attention because it strongly affects the structure of Figure 7.10. The constraint models for each discipline are constructed by first solving the optimization problem that minimizes the constraint violation with respect to local elastic and design variables.

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{n_s} s_{ik} \\
\text{with respect to} \quad & x_i, s_i \\
\text{subject to} \quad & c_i\left(x_0, x_i, y_i\left(x_0, x_i, y^t_{j\neq i}\right)\right) + s_i \geq 0 \\
& s_i \geq 0.
\end{aligned} \tag{7.13}$$

(Note that shared design variables and coupling targets are treated as parameters.) A post-optimality sensitivity analysis is then completed to determine the change in the optimized local design variables with respect to the change in shared design variables. Combining these post-optimality derivatives with the appropriate partial derivatives yields the linear constraint models. The optimized local design variables and elastic variables from Problem (7.13) are then used as part of the initial data for Problem (7.12). The full algorithm for ECO is listed in Algorithm 17.

Algorithm 17 Enhanced collaborative optimization

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate ECO iteration
repeat
for Each discipline i **do**
1: Create linear constraint model
2: Initiate disciplinary subproblem optimization
repeat
3: Interrogate nonlocal constraint models with local copies of shared variables
3.0: Evaluate disciplinary analysis
3.1: Compute disciplinary subproblem objective and constraints
4: Compute new disciplinary subproblem design point and J_i
until $4 \rightarrow 3$: Disciplinary optimization subproblem has converged
end for
5: Initiate system optimization
repeat
6: Compute J_0
7: Compute updated values of x_0 and y^t.
until $7 \rightarrow 6$: System optimization has converged
until $8 \rightarrow 1$: The J_0 is below specified tolerance

Based on the Roth's results [132, 133], ECO is effective in reducing the number of discipline analyses compared to CO. The trade-off is in the additional time required to build and update the models for each discipline, weighed against the simplified solution to the decomposed optimization problems. The results also show that ECO compares favorably with the Analytical Target Cascading architecture (which we will describe in Section 7.4.6).

While ECO seems to be effective, CO tends to be an inefficient architecture for solving MDO problems. Without any of the fixes discussed in this section, the architecture always requires a disproportionately large number of function and discipline evaluations [40, 83, 86, 167], assuming it converges at all. When the system-level equality constraints are relaxed, the results from CO are more competitive with other distributed architectures [107, 125, 155] but still compare poorly with the results of monolithic architectures.

7.4.5 Bilevel Integrated System Synthesis (BLISS)

The BLISS architecture [149], like CSSO, is a method for decomposing the MDF problem along disciplinary lines. Unlike CSSO, however, BLISS assigns local design variables to disciplinary subproblems and shared design variables to the system subproblem. The basic approach of the architecture is to form a path in the design space using a series of linear approximations to the original design problem, with user-defined bounds on the design variable steps, to prevent the design point from moving so far away that the approximations are too inaccurate. This is an idea similar to that of trust-region methods [37]. These approximations are constructed at each iteration using

global sensitivity information. The system level subproblem is formulated as

$$\begin{aligned} \text{minimize} \quad & (f_0^*)_0 + \left(\frac{\mathrm{d}f_0^*}{\mathrm{d}x_0}\right)\Delta x_0 \\ \text{with respect to} \quad & \Delta x_0 \\ \text{subject to} \quad & (c_0^*)_0 + \left(\frac{\mathrm{d}c_0^*}{\mathrm{d}x_0}\right)\Delta x_0 \geq 0 \\ & (c_i^*)_0 + \left(\frac{\mathrm{d}c_i^*}{\mathrm{d}x_0}\right)\Delta x_0 \geq 0 \quad \text{for } i = 1, \ldots, N \\ & \Delta x_{0L} \leq \Delta x_0 \leq \Delta x_{0U}. \end{aligned} \tag{7.14}$$

The discipline i subproblem is given by

$$\begin{aligned} \text{minimize} \quad & (f_0)_0 + \left(\frac{\mathrm{d}f_0}{\mathrm{d}x_i}\right)\Delta x_i \\ \text{with respect to} \quad & \Delta x_i \\ \text{subject to} \quad & (c_0)_0 + \left(\frac{\mathrm{d}c_0}{\mathrm{d}x_i}\right)\Delta x_i \geq 0 \\ & (c_i)_0 + \left(\frac{\mathrm{d}c_i}{\mathrm{d}x_i}\right)\Delta x_i \geq 0 \\ & \Delta x_{iL} \leq \Delta x_i \leq \Delta x_{iU}. \end{aligned} \tag{7.15}$$

Note the extra set of constraints in both system and discipline subproblems denoting the design variables bounds.

In order to prevent violation of the disciplinary constraints by changes in the shared design variables, post-optimality sensitivity information is required to solve the system subproblem. For this step, Sobieski [149] presents two methods: one based on a generalized version of the Global Sensitivity Equations [146], and another based on the "pricing" interpretation of local Lagrange multipliers. The resulting variants of BLISS are BLISS/A and BLISS/B, respectively. Other variations use response surface approximations to compute post-optimality sensitivity data [79, 84].

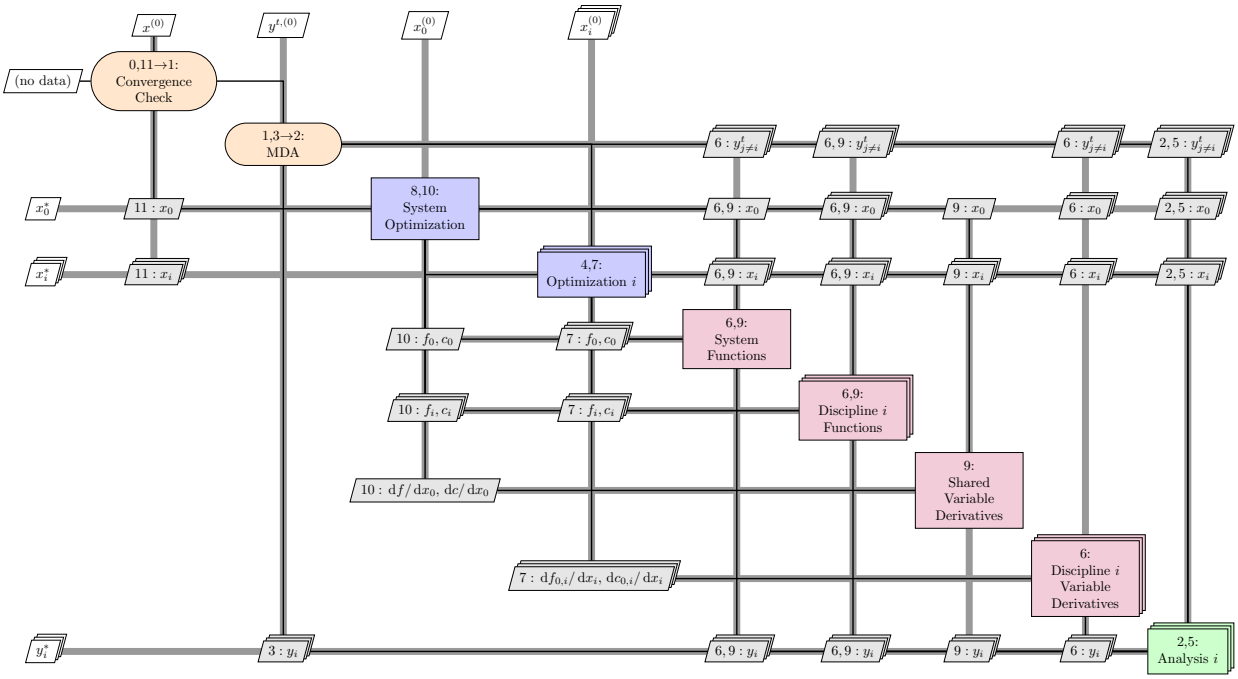

Figure 7.11: Diagram for the BLISS architecture

Algorithm 18 BLISS

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

0: Initiate system optimization
repeat
  1: Initiate MDA
  repeat
    2: Evaluate discipline analyses
    3: Update coupling variables
  until $3 \rightarrow 2$: MDA has converged
  4: Initiate parallel discipline optimizations
  for Each discipline i **do**
    5: Evaluate discipline analysis
    6: Compute objective and constraint function values and derivatives with respect to local design variables
    7: Compute the optimal solutions for the disciplinary subproblem
  end for
  8: Initiate system optimization
  9: Compute objective and constraint function values and derivatives with respect to shared design variables using post-optimality sensitivity analysis
  10: Compute optimal solution to system subproblem
until $11 \rightarrow 1$: System optimization has converged

Figure 7.11 shows the XDSM for BLISS and the procedure is listed in Algorithm 18. Note that

due to the linear nature of the optimization problems under consideration, repeated interrogation of the objective and constraint functions is not necessary once gradient information is available. However, this reliance on linear approximations is not without difficulties. If the underlying problem is highly nonlinear, the algorithm may converge slowly. The presence of user-defined variable bounds may help the convergence if these bounds are properly chosen, such as through a trust region framework. Detailed knowledge of the design space can also help, but this increases the overhead cost of implementation.

Two other adaptations of the original BLISS architecture are known in the literature. The first is Ahn and Kwon's proBLISS [2], an architecture for reliability-based MDO. Their results show that the architecture is competitive with reliability-based adaptations of MDF and IDF. The second is LeGresley and Alonso's BLISS/POD [96], an architecture that integrates a reduced-order modeling technique called Proper Orthogonal Decomposition [16] to reduce the cost of the multidisciplinary analysis and sensitivity analysis steps. Their results show a significant improvement in the performance of BLISS, to the point where it is almost competitive with MDF.

As an enhancement of the original BLISS, a radically different formulation called BLISS-2000 was developed by Sobieski et al. [150]. Because BLISS-2000 does not require a multidisciplinary analysis to restore feasibility of the design, we have separated it from other BLISS variants in the classification shown in Figure 7.7. In fact, like other IDF-derived architectures, BLISS-2000 uses coupling variable targets to enforce consistency at the optimum. Information exchange between system and discipline subproblems is completed through surrogate models of the disciplinary optima. The BLISS-2000 system subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, \tilde{y}\left(x, y^t\right)\right) \\ \text{with respect to} \quad & x_0, y^t, w \\ \text{subject to} \quad & c_0\left(x, \tilde{y}\left(x, y^t, w\right)\right) \geq 0 \\ & y_i^t - \tilde{y}_i\left(x_0, x_i, y_{j\neq i}^t, w_i\right) = 0 \quad \text{for } i = 1, \ldots, N. \end{aligned} \tag{7.16}$$

The BLISS-2000 discipline i subproblem is

$$\begin{aligned} \text{minimize} \quad & w_i^T y_i \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & c_i\left(x_0, x_i, y_i\left(x_0, x_i, y_{j\neq i}^t\right)\right) \geq 0. \end{aligned} \tag{7.17}$$

A unique aspect of this architecture is the use of a vector of weighting coefficients, w_i, attached to the disciplinary states. These weighting coefficients give the user a measure of control over state variable preferences. Generally speaking, the coefficients should be chosen based on the structure of the global objective to allow disciplinary subproblems to find an optimum more quickly. How much the choice of coefficients affects convergence has yet to be determined.

BLISS-2000 possesses several advantages over the original BLISS architecture. First, the solution procedure is much easier to understand. The XDSM for the architecture is shown in Figure 7.12 and the procedure is listed in Algorithm 19.

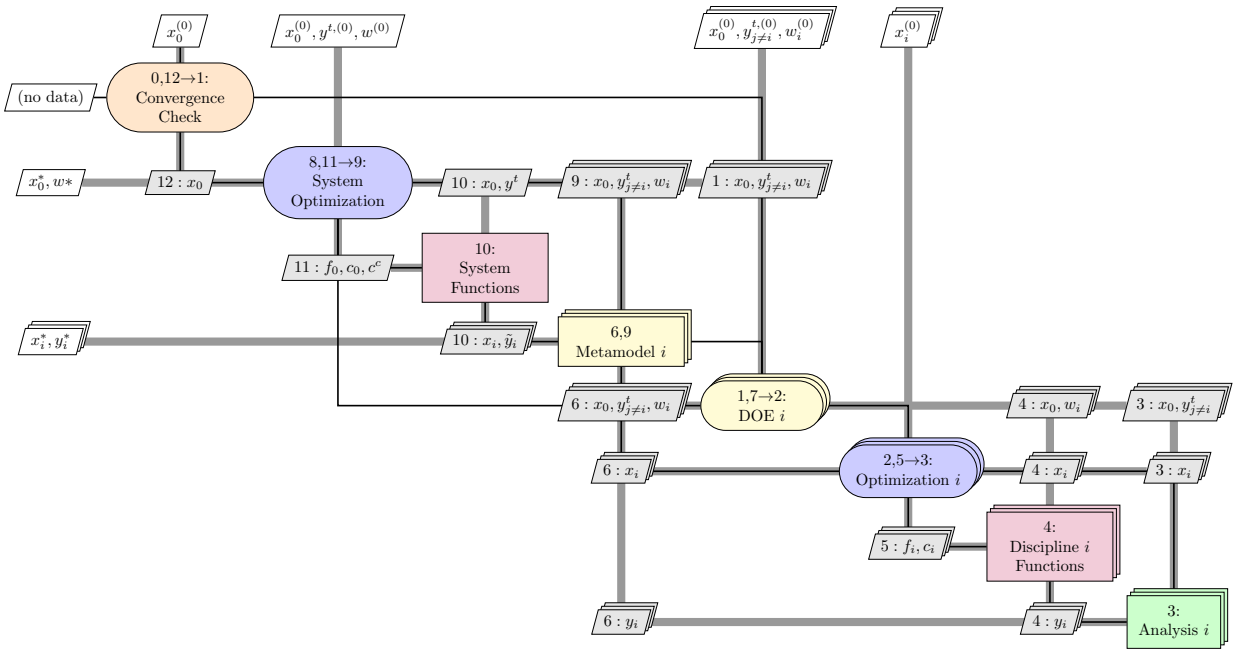

Figure 7.12: Diagram for the BLISS-2000 architecture

Algorithm 19 BLISS-2000

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

0: Initiate system optimization
repeat
  for Each discipline i **do**
    1: Initiate a DOE
    for Each DOE point **do**
      2: Initiate discipline subproblem optimization
      repeat
        3: Evaluate discipline analysis
        4: Compute discipline objective f_i and constraint functions c_i
        5: Update the local design variables x_i
      until $5 \rightarrow 3$: Disciplinary optimization subproblem has converged
      6: Update metamodel of optimized disciplinary subproblem with new solution
    end for $7 \rightarrow 1$
  end for
  8: Initiate system subproblem optimization
  repeat
    9: Interrogate metamodels with current values of system variables
    10: Compute system objective and constraint function values
    11: Compute a new system design point
  until $11 \rightarrow 9$ System subproblem has converged
until $12 \rightarrow 1$: System optimization has converged

Such simplicity greatly reduces the number of obstacles to implementation. Second, the decomposed problem formulation of BLISS-2000 is equivalent to Problem (7.1) [150]. Third, by using metamodels for each discipline, rather than for the whole system, the calculations for BLISS-2000 can be run in parallel with minimal communication between disciplines. Finally, BLISS-2000 seems to be more flexible than its predecessor. Recently, Sobieski detailed an extension of BLISS-2000 to handle multilevel, system-of-systems problems [147]. In spite of these advantages, it appears that BLISS-2000 has not been used nearly as frequently as the original BLISS formulation.

Most of the benchmarking of BLISS used its the original formulation. Many of the results available [40, 125, 167] suggest that BLISS is not competitive with other architectures in terms of computational cost. Concerns include both the number of discipline analyses required and the high cost of the post-optimality sensitivity computations. These problems can be mitigated by the use of surrogate models or reduced-order modeling as described above. The one benchmarking result available for BLISS-2000 is the launch vehicle design problem of Brown and Olds [27]. In this case, BLISS-2000 tends to outperform other distributed architectures and should be cost-competitive with other monolithic architectures when coarse-grained parallel processing is fully exploited. While this is a promising development, more work is needed to confirm the result.

7.4.6 Analytical Target Cascading (ATC)

The ATC architecture was not initially developed as an MDO architecture, but as a method to propagate system targets — i.e., requirements or desirable properties — through a hierarchical system to achieve a feasible system design satisfying these targets [80, 81]. If the system targets were unattainable, the ATC architecture would return a design point minimizing the inattainability. Effectively, then, the ATC architecture is no different from an MDO architecture with a system objective of minimizing the squared difference between a set of system targets and model responses. By simply changing the objective function, we can solve general MDO problems using ATC.

The ATC problem formulation that we present here is due to Tosserams et al. [158]. This formulation conforms to our definition of an MDO problem by explicitly including system wide objective and constraint functions. Like all other ATC problem formulations, it is mathematically equivalent to Problem (7.1). The ATC system subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y^t\right) + \sum_{i=1}^{N} \Phi_i\left(\hat{x}_{0i} - x_0, y_i^t - y_i\left(x_0, x_i, y^t\right)\right) + \\ & \Phi_0\left(c_0\left(x, y^t\right)\right) \\ \text{with respect to} \quad & x_0, y^t, \end{aligned} \tag{7.18}$$

where Φ_0 is a penalty relaxation of the global design constraints and Φ_i is a penalty relaxation of the discipline i consistency constraints. The i^{th} discipline subproblem is:

$$\begin{aligned} \text{minimize} \quad & f_0\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y_{j\neq i}^t\right), y_{j\neq i}^t\right) + f_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y_{j\neq i}^t\right)\right) + \\ & \Phi_i\left(y_i^t - y_i\left(\hat{x}_{0i}, x_i, y_{j\neq i}^t\right), \hat{x}_{0i} - x_0\right) + \\ & \Phi_0\left(c_0\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y_{j\neq i}^t\right), y_{j\neq i}^t\right)\right) \\ \text{with respect to} \quad & \hat{x}_{0i}, x_i \\ \text{subject to} \quad & c_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y_{j\neq i}^t\right)\right) \geq 0. \end{aligned} \tag{7.19}$$

Figure 7.13 shows the ATC architecture XDSM, where w denotes the penalty function weights used in the determination of Φ_0 and Φ_i. The details of ATC are described in Algorithm 20.

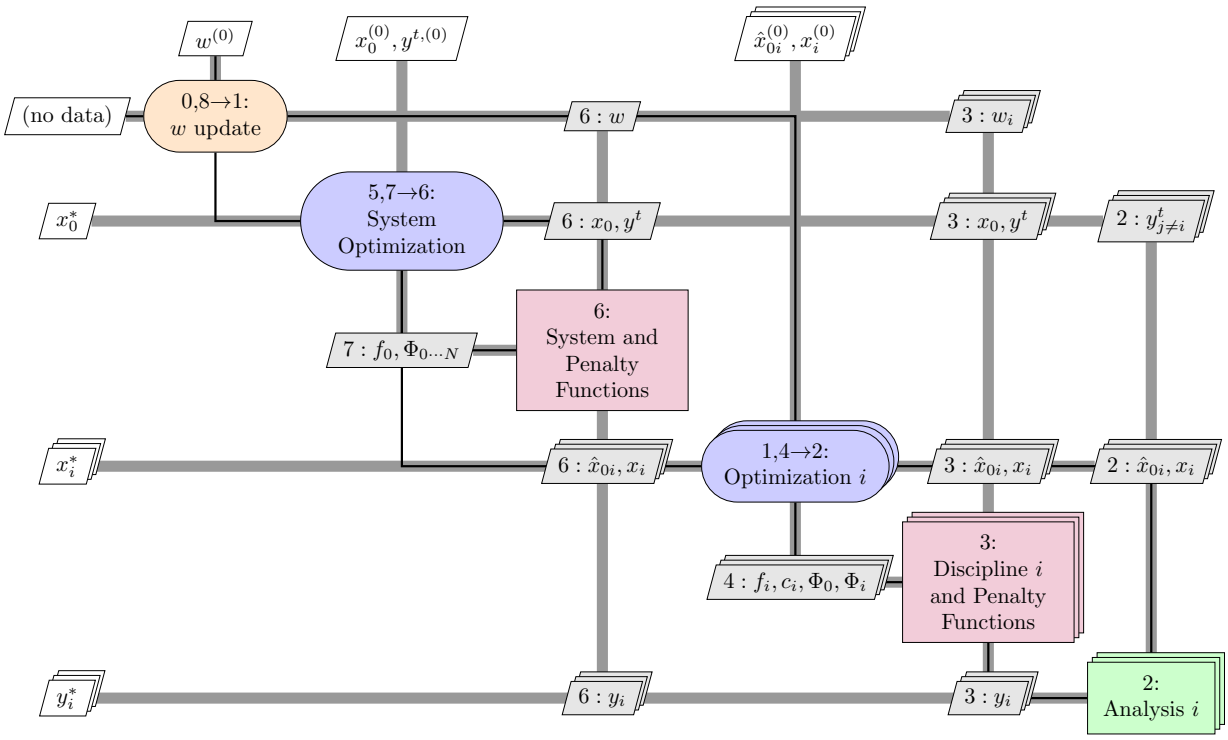

Figure 7.13: Diagram for the ATC architecture

Algorithm 20 ATC

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

0: Initiate main ATC iteration
repeat
  for Each discipline i **do**
    1: Initiate discipline optimizer
    repeat
      2: Evaluate disciplinary analysis
      3: Compute discipline objective and constraint functions and penalty function values
      4: Update discipline design variables
    until $4 \rightarrow 2$: Discipline optimization has converged
  end for
  5: Initiate system optimizer
  repeat
    6: Compute system objective, constraints, and all penalty functions
    7: Update system design variables and coupling targets.
  until $7 \rightarrow 6$: System optimization has converged
  8: Update penalty weights
until $8 \rightarrow 1$: Penalty weights are large enough

Note that ATC can be applied to a multilevel hierarchy of systems just as well as a discipline-

based non-hierarchic system. In the multilevel case, the penalty functions are applied to all constraints that combine local information with information from the levels immediately above or below the current one. Also note that post-optimality sensitivity data is not needed in any of the subproblems as nonlocal data are always treated as fixed values in the current subproblem.

The most common penalty functions in ATC are quadratic penalty functions. In this case, the proper selection of the penalty weights is important for both final inconsistency in the discipline models and convergence of the algorithm. Michalek and Papalambros [115] present an effective weight update method that is especially useful when unattainable targets have been set in a traditional ATC process. Michelena et al. [118] present several coordination algorithms using ATC with quadratic penalty functions and demonstrate the convergence for all of them. Note that, as with penalty methods for general optimization problems, (see, e.g., Nocedal and Wright [119, chap. 17]) the solution of the MDO problem must be computed to reasonable accuracy before the penalty weights are updated. However, because we are now dealing with a distributed set of subproblems, the whole hierarchy of subproblems must be solved for a given set of weights. This is due to the nonseparable nature of the quadratic penalty function.

Several other penalty function choices and associated coordination approaches have also been devised for ATC. Kim et al. [78] outline a version of ATC that uses Lagrangian relaxation and a sub-gradient method to update the multiplier estimates. Tosserams et al. [156] use augmented Lagrangian relaxation with Bertsekas' method of multipliers [17] and alternating direction method of multipliers [18] to update the penalty weights. They also group this variant of ATC into a larger class of coordination algorithms known as Augmented Lagrangian Coordination [158]. Li et al. [98] apply the diagonal quadratic approximation approach of Ruszcynski [134] to the augmented Lagrangian to eliminate subproblem coupling through the quadratic terms and further parallelize the architecture. Finally, Han and Papalambros [65] propose a version of ATC based on sequential linear programming [14, 31], where inconsistency is penalized using infinity norms. They later presented a convergence proof of this approach in a short note [66]. For each of the above penalty function choices, ATC was able to produce the same design solutions as the monolithic architectures.

Despite having been developed relatively recently, the ATC architecture has been widely used. By far, ATC has been most frequently applied to design problems in the field for which it was developed, the automotive industry [22, 32, 67, 77, 82, 88, 89, 152]. However, the ATC approach has also proven to be useful in aircraft design [8, 9, 163] and building design [35]. ATC has also found applications outside of strict engineering design problems, including manufacturing decisions [99], supply chain management [70], and marketing decisions in product design [117]. Huang et al. [71] have developed an ATC-specific web portal to solve optimization problems via the ATC architecture. Etman et al. [46] discuss the automatic implementation of coordination procedures, using ATC as an example architecture. We also note the presence of ATC formulations that can handle integer variables [116] and probabilistic design problems [90, 102]. Another important adaptation of ATC applies to problems with block-separable linking constraints [160]. In this class of problems, c_0 consists of constraints which are sums of functions depending on only shared variables and the local variables of one discipline.

The performance of ATC compared with other architectures is not well known because only one result is available. In de Wit and Van Keulen's architecture comparison [40], ATC is competitive with all other benchmarked distributed architectures, including standard versions of CSSO, CO, and BLISS. However, ATC and the other distributed architectures are not competitive with a monolithic architecture in terms of the number of function and discipline evaluations. More commonly, different versions of ATC are benchmarked against each other. Tosserams et al. [156] compared the augmented Lagrangian penalty approach with the quadratic penalty approach and found much improved results with the alternating direction method of multipliers. Surprisingly,

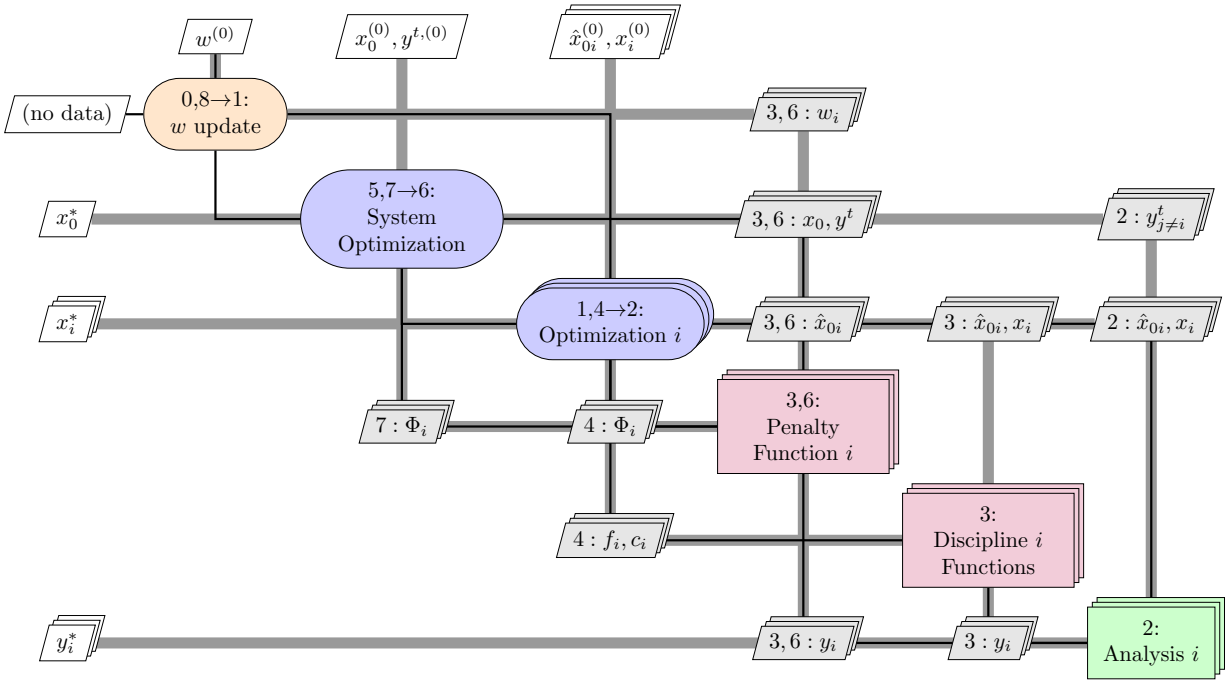

Figure 7.14: Diagram for the penalty decomposition architectures EPD and IPD

de Wit and van Keulen [40] found the augmented Lagrangian version performed worse than the quadratic penalty method for their test problem. Han and Papalambros [65] compared their sequential linear programming version of ATC to several other approaches and found a significant reduction in the number of function evaluations. However, they note that the coordination overhead is large compared to other ATC versions and still needs to be addressed.

7.4.7 Exact and Inexact Penalty Decomposition (EPD and IPD)

If there are no system-wide constraints or objectives, i.e., if neither f_0 and c_0 exist, the Exact or Inexact Penalty Decompositions (EPD or IPD) [42, 44] may be employed. Both formulations rely on solving the disciplinary subproblem

$$\begin{aligned} \text{minimize} \quad & f_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, y^t_{j\neq i}\right)\right) + \Phi_i\left(\hat{x}_{0i} - x_0, y_i^t - y_i\left(\hat{x}_{0i}, x_i, y^t_{j\neq i}\right)\right) \\ \text{with respect to} \quad & \hat{x}_{0i}, x_i \\ \text{subject to} \quad & c_i\left(\hat{x}_{0i}, x_i, y_i\left(x_{0i}, x_i, y^t_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{7.20}$$

Here, Φ_i denotes the penalty function associated with the inconsistency between the i^{th} disciplinary information and the system information. In EPD, Φ_i is an L_1 penalty function with additional variables and constraints added to ensure smoothness. In IPD, Φ_i is a quadratic penalty function with appropriate penalty weights. The notation $\hat{x}_{0i}$ denotes a local copy of the shared design variables in discipline i, while x_0 denotes the system copy.

At the system level, the subproblem is an unconstrained minimization with respect to the target variables. The objective function is the sum of the optimized disciplinary penalty terms, denoted

as Φ_i^*.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} \Phi_i^* \left(x_0, y^t\right) = \sum_{i=1}^{N} \Phi_i \left(\hat{x}_{0i} - x_0, y_i^t - y_i\left(\hat{x}_{0i}, x_i, y_{j\neq i}^t\right)\right) \\ \text{with respect to} \quad & x_0, y^t \end{aligned} \tag{7.21}$$

The penalty weights are updated upon solution of the system problem. Figure 7.14 shows the XDSM for this architecture, where w represents the penalty weights. The sequence of operations in this architecture is detailed in Algorithm 21.

Algorithm 21 EPD and IPD

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate main iteration
repeat
  for Each discipline i **do**
    repeat
      1: Initiate discipline optimizer
      2: Evaluate discipline analysis
      3: Compute discipline objective and constraint functions, and penalty function values
      4: Update discipline design variables
    until $4 \rightarrow 2$: Discipline optimization has converged
  end for
  5: Initiate system optimizer
  repeat
    6: Compute all penalty functions
    7: Update system design variables and coupling targets
  until $7 \rightarrow 6$: System optimization has converged
  8: Update penalty weights.
until $8 \rightarrow 1$: Penalty weights are large enough

Both EPD and IPD have mathematically provable convergence under the linear independence constraint qualification and with mild assumptions on the update strategy for the penalty weights [44]. In particular, the penalty weight in IPD must monotonically increase until the inconsistency is sufficiently small, similar to other quadratic penalty methods [119]. For EPD, the penalty weight must be larger than the largest Lagrange multiplier, following established theory of the L_1 penalty function [119], while the barrier parameter must monotonically decrease like in an interior point method [165]. If other penalty functions are employed, the parameter values are selected and updated according to the corresponding mathematical theory. Furthermore, under these conditions, the solution obtained under EPD and IPD will also be a solution to Problem (7.1).

Only once in the literature has either penalty decomposition architecture been tested against any others. The results of Tosserams et al. [157] suggest that performance depends on the choice of penalty function employed. A comparison between IPD with a quadratic penalty function and IPD with an augmented Lagrangian penalty function showed that the latter significantly outperformed the former in terms of both time and number of function evaluations on several test problems.

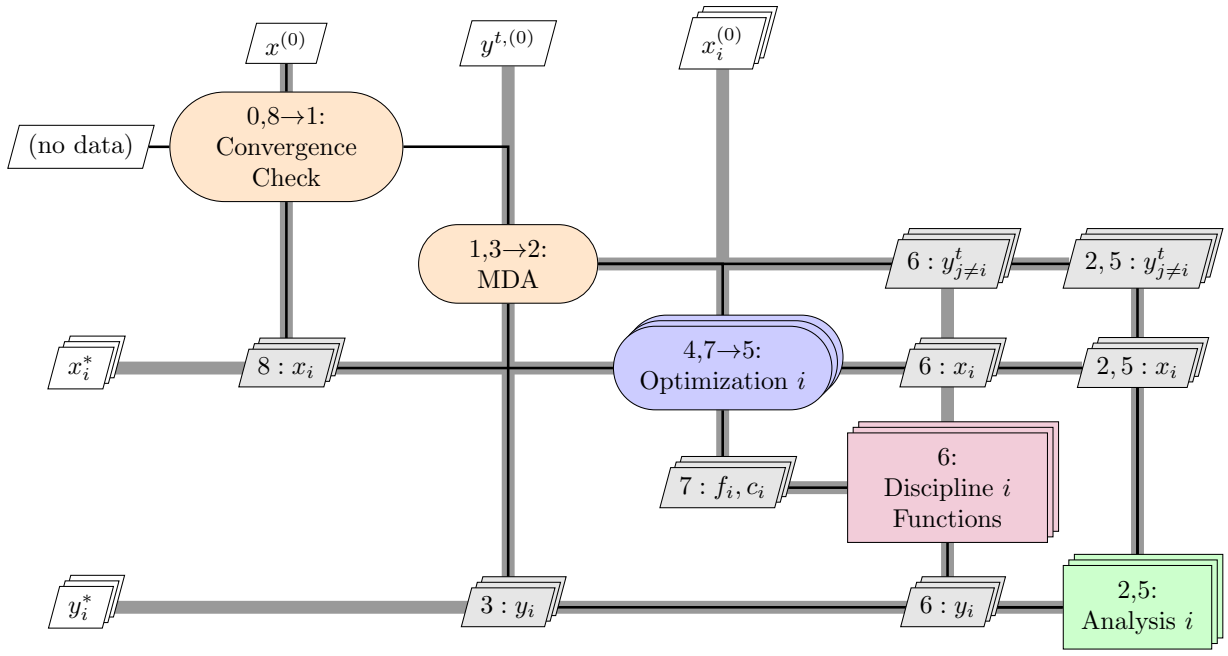

Figure 7.15: Diagram for the MDOIS architecture

7.4.8 MDO of Independent Subspaces (MDOIS)

If the problem contains no system-wide constraints or objectives, i.e., if neither f_0 and c_0 exist, and the problem does not include shared design variables, i.e., if x_0 does not exist, then the MDO of independent subspaces (MDOIS) architecture [141] applies. In this case, the discipline subproblems are fully separable (aside from the coupled state variables) and given by

$$\begin{aligned} \text{minimize} \quad & f_i\left(x_i, y_i\left(x_i, y^t_{j\neq i}\right)\right) \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & c_i\left(x_i, y_i\left(x_i, y^t_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{7.22}$$

In this case, the targets are just local copies of system state information. Upon solution of the disciplinary problems, which can access the output of individual disciplinary analysis codes, a full multidisciplinary analysis is completed to update all target values. Thus, rather than a system subproblem used by other architectures, the MDA is used to guide the disciplinary subproblems to a design solution. Shin and Park [141] show that under the given problem assumptions an optimal design is found using this architecture. Figure 7.15 depicts this process in an XDSM. MDOIS is detailed in Algorithm 22.

Algorithm 22 MDOIS

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

```
0: Initiate main iteration
repeat
  repeat
    1: Initiate MDA
    2: Evaluate discipline analyses
    3: Update coupling variables
  until 3 → 2: MDA has converged
  for Each discipline i do
    4: Initiate disciplinary optimization
    repeat
      5: Evaluate discipline analysis
      6: Compute discipline objectives and constraints
      7: Compute a new discipline design point
    until 7 → 5: Discipline optimization has converged
  end for
until 8 → 1 Main iteration has converged
```

Benchmarking results are available comparing MDOIS to some of the older architectures. These results are given by Yi et al. [167]. In many cases, MDOIS requires fewer analysis calls than MDF while still being able to reach the optimal solution. However, MDOIS still does not converge as fast as IDF and the restrictive definition of the problem means that the architecture is not nearly as flexible as MDF. A practical problem that can be solved using MDOIS is the belt-integrated seat problem of Shin et al. [140]. However, the results using MDOIS have not been compared to results obtained using other architectures.

7.4.9 Quasiseparable Decomposition (QSD)

Haftka and Watson [63] developed the QSD architecture to solve *quasiseparable* optimization problems. In a quasiseparable problem, the system objective and constraint functions are assumed to be dependent only on global variables (i.e., the shared design and coupling variables). This type of problem may be thought of as identical to the complicating variables problems discussed in Section 7.4.1. In our experience, we have not come across any practical design problems that do not satisfy this property. However, if required by the problem, we can easily transform the general MDO problem (7.1) into a quasiseparable problem. This is accomplished by duplicating the relevant local variables, and forcing the global objective to depend on the target copies of local variables. (The process is analogous to adapting the general MDO problem to the original Collaborative Optimization architecture.) The resulting quasiseparable problem and decomposition is mathematically equivalent to the original problem. The system subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x_0, y^t\right) + \sum_{i=1}^{N} b_i \\ \text{with respect to} \quad & x_0, y^t, b \\ \text{subject to} \quad & c_0\left(x_0, y^t\right) \geq 0 \\ & s_i^*\left(x_0, x_i, y_i\left(x_0, x_i, y_{j\neq i}^t\right), b_i\right) \geq 0 \quad \text{for } i = 1, \ldots, N. \end{aligned} \tag{7.23}$$

where s_i is the constraint margin for discipline i and b_i is the "budget" assigned to each disciplinary objective. The discipline i subproblem becomes

$$\begin{aligned} \text{minimize} \quad & -s_i \\ \text{with respect to} \quad & x_i, s_i \\ \text{subject to} \quad & c_i\left(x_0, x_i, y_i\left(x_0, x_i, y^t_{j\neq i}\right)\right) - s_i \geq 0 \\ & f_i\left(x_0, x_i, y_i\left(x_0, x_i, y^t_{j\neq i}\right)\right) - b_i - s_i \geq 0 \\ & y^t_i - y_i\left(x_0, x_i, y^t_{j\neq i}\right) = 0 \end{aligned} \tag{7.24}$$

where k is an element of the constraint vector c_i. Due to the use of target copies, we classify this architecture as distributed IDF. Note that, like CO, this is a bilevel architecture where the solutions of disciplinary subproblems are constraints in the system subproblem. Therefore, post-optimality sensitivities or surrogate model approximations of optimized disciplinary subproblems are required to solve the system subproblem. The standard architecture is shown as an XDSM in Figure 7.16. The sequence of operations in QSD is given by Algorithm 23.

Algorithm 23 QSD

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*
0: Initiate system optimization
repeat
1: Compute system objectives and constraints
for Each discipline i **do**
1.0: Initiate discipline optimization
repeat
1.1: Evaluate discipline analysis
1.2: Compute discipline objective and constraints
1.3: Update discipline design point
until 1.3 $\rightarrow$ 1.1: Discipline optimization has converged
end for
2: Compute a new system design point
until 2 $\rightarrow$ 1: System problem has converged

We also note that Haftka and Watson have extended the theory behind QSD to solve problems with a combination of discrete and continuous variables [64].

Liu et al. [101] successfully applied QSD with surrogate models to a structural optimization problem. However, they made no comparison of the performance to other architectures, not even QSD without the surrogates. A version of QSD without surrogate models was benchmarked by de Wit and van Keulen [40]. Unfortunately, this architecture was the worst of all the architectures tested in terms of disciplinary evaluations. A version of QSD using surrogate models should yield improved performance, due to the smoothness introduced by the model, but this version has not been benchmarked to our knowledge.

7.4.10 Asymmetric Subspace Optimization (ASO)

The ASO architecture [34] is a new distributed-MDF architecture. It was motivated by the case of high-fidelity aerostructural optimization, where the aerodynamic analysis typically requires an

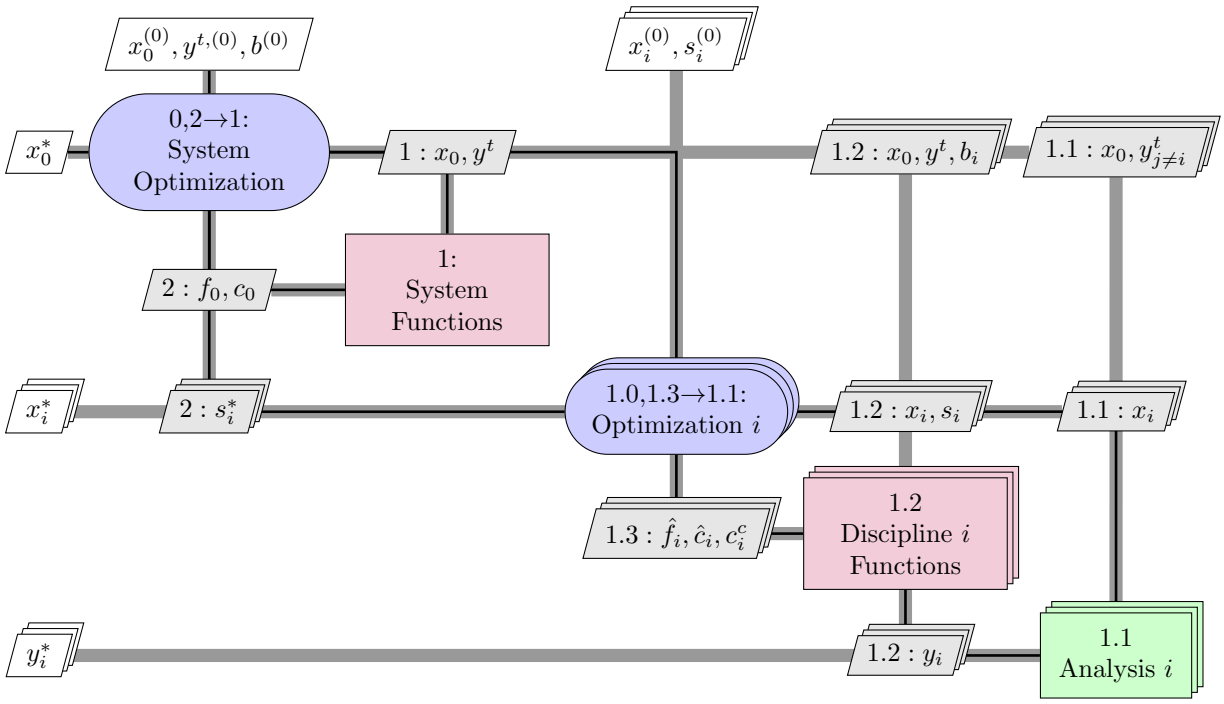

Figure 7.16: Diagram for the QSD architecture

order of magnitude more time to complete than the structural analysis [110]. To reduce the number of expensive aerodynamic analyses, the structural analysis is coupled with a structural optimization inside the MDA. This idea can be readily generalized to any problem where there is a wide discrepancy between discipline analysis times. Figure 7.17 shows the optimization of the third discipline of a generic problem within the ASO architecture. The sequence of operations in ASO is listed in Algorithm 24.

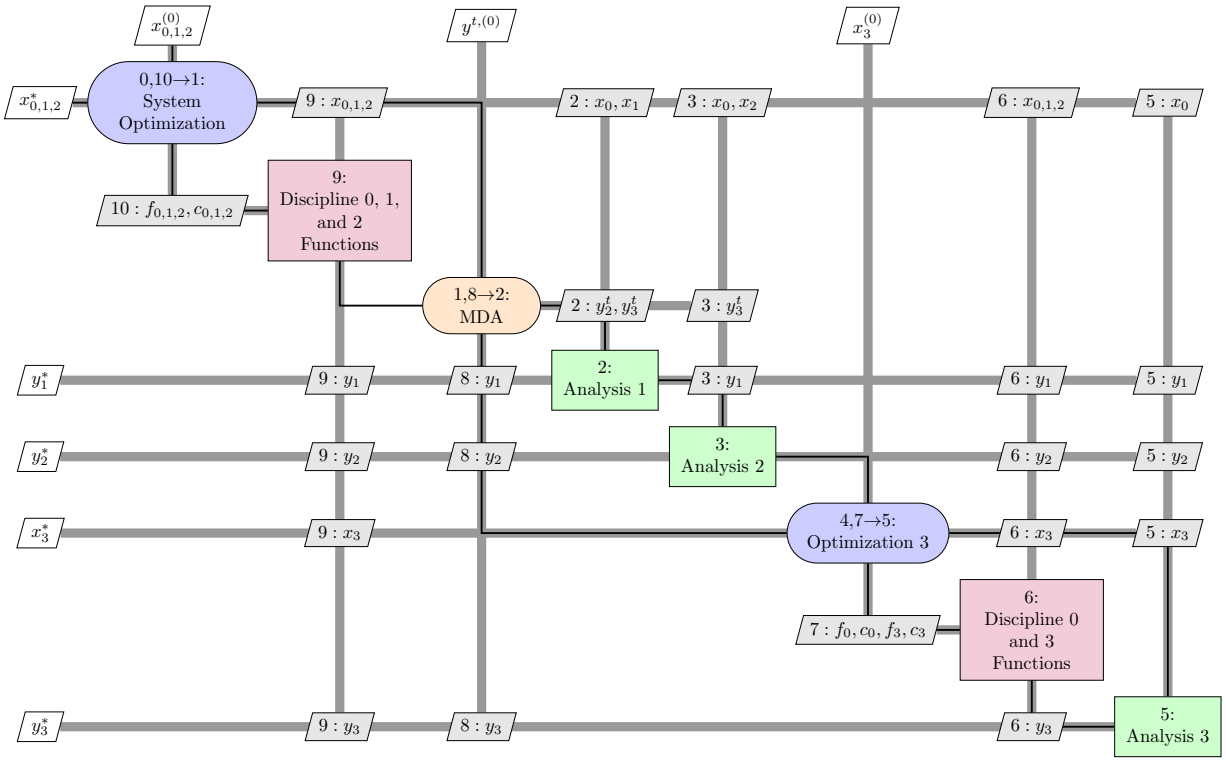

Figure 7.17: Diagram for the ASO architecture

Algorithm 24 ASO

Input: Initial design variables x
Output: Optimal variables x^*, objective function f^*, and constraint values c^*

```
0: Initiate system optimization
repeat
  1: Initiate MDA
  repeat
    2: Evaluate Analysis 1
    3: Evaluate Analysis 2
    4: Initiate optimization of Discipline 3
    repeat
      5: Evaluate Analysis 3
      6: Compute discipline 3 objectives and constraints
      7: Update local design variables
    until 7 → 5: Discipline 3 optimization has converged
    8: Update coupling variables
  until 8 → 2 MDA has converged
  9: Compute objective and constraint function values for all disciplines 1 and 2
  10: Update design variables
until 10 → 1: System optimization has converged
```

The optimality of the final solution is preserved by using the coupled post-optimality sensitivity (CPOS) equations, developed by Chittick and Martins [34], to calculate gradients at the system level. CPOS represents the extension of the coupled sensitivity equations [111, 148] to include the optimality conditions. ASO was later implemented using a coupled-adjoint approach as well [33]. Kennedy et al. [76] present alternative strategies for computing the disciplinary subproblem optima and the post-optimality sensitivity analysis. As with other bilevel MDO architectures, the post-optimality analysis is necessary to ensure convergence to an optimal design of the original monolithic problem.

The system subproblem in ASO is

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\left(x,y\right)\right) + \sum_k f_k\left(x_0, x_k, y_k\left(x_0, x_k, y_{j\neq k}\right)\right) \\ \text{with respect to} \quad & x_0, x_k \\ \text{subject to} \quad & c_0\left(x, y\left(x,y\right)\right) \geq 0 \\ & c_k\left(x_0, x_k, y_k\left(x_0, x_k, y_{j\neq k}\right)\right) \geq 0 \qquad \text{for all } k, \end{aligned} \tag{7.25}$$

where subscript k denotes disciplinary information that remains outside of the MDA. The disciplinary problem for discipline i, which is resolved inside the MDA, is

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\left(x,y\right)\right) + f_i\left(x_0, x_i, y_i\left(x_0, x_i, y_{j\neq i}\right)\right) \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & c_i\left(x_0, x_i, y_i\left(x_0, x_i, y_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{7.26}$$

The results show a substantial reduction in the number of calls to the aerodynamics analysis, and even a slight reduction in the number of calls to the structural analysis [34]. However, the total time for the optimization routine is only competitive with MDF if the aerodynamic analysis is substantially more costly than the structural analysis. If the two analyses take roughly equal

time, MDF is still much faster. Furthermore, the use of CPOS increases the complexity of the sensitivity analysis compared to a normal coupled adjoint, which adds to the overall solution time. As a result, this architecture may only appeal to practitioners solving MDO problems with widely varying computational cost in the discipline analysis.

7.5 Architecture Benchmarking Issues

One of the challenges facing the MDO community is determining which architecture is most efficient for a given MDO problem. One way this can be accomplished is by benchmarking the architectures using simple test problems. Many authors developing new architectures include a few results in their work. However, it is especially important to test multiple architectures on a given problem, as the most appropriate architecture often depends on the nature of the problem itself. A number of studies specifically focused on benchmarking are available in the literature [10, 27, 73, 83, 86, 125, 155, 167]. However, there are a number of limitations in these studies that need to be considered when reading them and considering future work.

The first limitation is the fact that no two studies are likely to produce identical results. This stems from the fact that no two programmers can be expected to program the same architecture the same way. In addition, experts in one particular architecture may be able to more fully optimize the performance of their architecture, unintentionally biasing the results [167]. This problem can be overcome by performing the benchmarking in a specialized MDO framework such as iSight, ModelCenter, pyMDO [112], or OpenMDAO [53]. These frameworks allow single definitions of the MDO problem and the architecture to be frequently reused, eliminating most of the "human factor" from the optimization. Padula and Gillian [120] present a brief history of frameworks for MDO. Kodiyalam and Sobieski [85] and Kodiyalam et al. [87] discuss requirements for an MDO framework and high-performance computing considerations respectively.

The second limitation is the choice of architectures used in the benchmarking studies. Most studies tend to focus on the most mature architectures, especially MDF, IDF, CO, and CSSO. This should become less of a limitation with time as the newer architectures become better known. We must emphasize, though, that implementing the new architecture in an established MDO framework should allow more rapid benchmarking of the new architecture and give a much earlier indication of the architecture's promise.

The third limitation is in the test problems themselves. Almost all of the test problems are of low dimensionality, and many have discipline analyses consisting of analytic functions. In high-fidelity design, the MDO problem could have thousands of variables and constraints with discipline analyses that take minutes or hours to evaluate, even using high-performance parallel computers. While the test problems used in benchmarking may never reach this level of complexity, they should be large enough to establish a clear trend in terms of time and number of function calls. Ideally, these problems would also be scalable to examine the effects of problem dimensionality on architecture efficiency. An early attempt at such a scalable problem was presented by Tedford and Martins [155] and allowed the user to define an arbitrary number of disciplines, design variables, and coupling variables. Another benchmarking problem that possesses some scalability is the microaccelerometer problem of Tosserams et al. [162] Test problems with lower dimensionality but with multiple local minima are presented by Tosserams et al. [161] We also feel that creating a test suite of MDO problems similar to the now-defunct NASA MDO suite [121] would be of great utility to researchers and practitioners in the field.

Bibliography

[1] Jeremy Agte, Olivier de Weck, Jaroslaw Sobieszczanski-Sobieski, Paul Arendsen, Alan Morris, and Martin Spieck. MDO: Assessment and direction for advancement — an opinion of one international group. *Structural and Multidisciplinary Optimization*, 40:17–33, 2010. doi: 10.1007/s00158-009-0381-5.

[2] J Ahn and J H Kwon. An Efficient Strategy for Reliability-Based Multidisciplinary Design Optimization using BLISS. *Structural and Multidisciplinary Optimization*, 31:363–372, 2006. doi: 10.1007/s00158-005-0565-6.

[3] Natalia Alexandrov and M. Y. Hussaini, editors. *Multidisciplinary Design Optimization: State-of-the-Art*. SIAM, 1997.

[4] Natalia M. Alexandrov. Multilevel methods for MDO. In Natalia M. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the-Art*. SIAM, Philadelphia, 1997.

[5] Natalia M Alexandrov and Robert Michael Lewis. Comparative Properties of Collaborative Optimization and Other Approaches to MDO. In *1st ASMO UK/ISSMO Conference on Engineering Design Optimization*, 1999.

[6] Natalia M Alexandrov and Robert Michael Lewis. Analytical and Computational Aspects of Collaborative Optimization for Multidisciplinary Design. *AIAA Journal*, 40(2):301–309, 2002. doi: 10.2514/2.1646.

[7] Natalia M Alexandrov and Robert Michael Lewis. Reconfigurability in MDO Problem Synthesis, Part 1. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, number September, Albany, NY, 2004.

[8] James T Allison, Brian Roth, Michael Kokkolaras, Ilan M Kroo, and Panos Y Papalambros. Aircraft Family Design Using Decomposition-Based Methods. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, September 2006.

[9] James T Allison, David Walsh, Michael Kokkolaras, Panos Y Papalambros, and Matthew Cartmell. Analytical Target Cascading in Aircraft Design. In *44th AIAA Aerospace Sciences Meeting*, 2006.

[10] James T Allison, Michael Kokkolaras, and Panos Y Papalambros. On Selecting Single-Level Formulations for Complex System Design Optimization. *Journal of Mechanical Design*, 129 (September):898–906, September 2007. doi: 10.1115/1.2747632.

[11] R Balling and M R Rawlings. Collaborative Optimization with Disciplinary Conceptual Design. *Structural and Multidisciplinary Optimization*, 20(3):232–241, November 2000. doi: 10.1007/s001580050151.

[12] Richard J. Balling. Approaches to MDO which support disciplinary autonomy. In Natalia M. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the-Art*, pages 90–97. SIAM, 1997.

[13] Richard J Balling and Jaroslaw Sobieszczanski-Sobieski. Optimization of Coupled Systems: A Critical Overview of Approaches. *AIAA Journal*, 34(1):6–17, 1996. doi: 10.2514/3.13015.

[14] M S Bazaara, H D Sherali, and C M Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 2006.

[15] J F Benders. Partitioning Procedures for Solving Mixed Variables Programming Problems. *Numerische Mathematik*, 4:238–252, 1962. doi: 10.1007/BF01386316.

[16] Gal Berkooz, Philip Holmes, and John L Lumley. The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. *Annual Review of Fluid Mechanics*, 25:539–575, January 1993. doi: 10.1146/annurev.fl.25.010193.002543.

[17] Dimitri P Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1996.

[18] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[19] Lorentz T Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders, editors. *Large-Scale PDE-Constrained Optimization*. Springer-Verlag, 2003.

[20] C L Bloebaum, P Hajela, and J Sobieszczanski-Sobieski. Non-Hierarchic System Decomposition in Structural Optimization. *Engineering Optimization*, 19(3):171–186, 1992. doi: 10.1080/03052159208941227.

[21] C.L. Bloebaum. Coupling strength-based system reduction for complex engineering design. *Structural Optimization*, 10:113–121, 1995.

[22] V Y Blouin, G M Fadel, I U Haque, J R Wagner, and H B Samuels. Continuously Variable Transmission Design for Optimum Vehicle Performance by Analytical Target Cascading. *International Journal of Heavy Vehicle Systems*, 11:327–348, 2004. doi: 10.1504/IJHVS.2004.005454.

[23] R D Braun, A A Moore, and I M Kroo. Collaborative Approach to Launch Vehicle Design. *Journal of Spacecraft and Rockets*, 34(4):478–486, July 1997. doi: 10.2514/2.3237.

[24] Robert D Braun. *Collaborative Optimization: An Architecture for Large-Scale Distributed Design*. PhD thesis, Stanford University, Stanford, CA 94305, 1996.

[25] Robert D Braun and Ilan M Kroo. Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment. In N Alexandrov and M Y Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the-Art*, pages 98–116. SIAM, 1997.

[26] Robert D Braun, Peter Gage, Ilan M Kroo, and Ian P Sobieski. Implementation and Performance Issues in Collaborative Optimization. In *6th AIAA, NASA, and ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1996.

[27] Nichols F Brown and John R Olds. Evaluation of Multidisciplinary Optimization Techniques Applied to a Reusable Launch Vehicle. *Journal of Spacecraft and Rockets*, 43(6):1289–1300, 2006. doi: 10.2514/1.16577.

[28] Tyson R Browning. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management*, 48(3):292–306, 2001. doi: 10.1109/17.946528.

[29] Irene A Budianto and John R Olds. Design and Deployment of a Satellite Constellation Using Collaborative Optimization. *Journal of Spacecraft and Rockets*, 41(6):956–963, November 2004. doi: 10.2514/1.14254.

[30] Guobiao Cai, Jie Fang, Yuntao Zheng, Xiaoyan Tong, Jun Chen, and Jue Wang. Optimization of System Parameters for Liquid Rocket Engines with Gas-Generator Cycles. *Journal of Propulsion and Power*, 26(1):113–119, 2010. doi: 10.2514/1.40649.

[31] Ting-Yu Chen. Calculation of the Move Limits for the Sequential Linear Programming Method. *International Journal for Numerical Methods in Engineering*, 36(15):2661–2679, August 1993. doi: 10.1002/nme.1620361510.

[32] Yong Chen, Xiaokai Chen, and Yi Lin. The Application of Analytical Target Cascading in Parallel Hybrid Electric Vehicle. In *IEEE Vehicle Power and Propulsion Conference*, pages 1602–1607, 2009. ISBN 9781424426010.

[33] Ian R Chittick and Joaquim R R A Martins. Aero-Structural Optimization Using Adjoint Coupled Post-Optimality Sensitivities. *Structural and Multidisciplinary Optimization*, 36: 59–70, 2008. doi: 10.1007/s00158-007-0200-9.

[34] Ian R. Chittick and Joaquim R. R. A. Martins. An asymmetric suboptimization approach to aerostructural optimization. *Optimization and Engineering*, 10(1):133–152, March 2009. doi: 10.1007/s11081-008-9046-2.

[35] R Choudhary, A Malkawi, and P Y Papalambros. Analytic Target Cascading in Simulation-based Building Design. *Automation in Construction*, 14(4):551–568, August 2005. doi: 10.1016/j.autcon.2004.11.004.

[36] A J Conejo, F J Nogales, and F J Prieto. A Decomposition Procedure Based on Approximate Newton Directions. *Mathematical Programming*, 93:495–515, 2002.

[37] Andrew R Conn, Nicholas I M Gould, and Philippe L Toint. *Trust Region Methods*. SIAM, Philadelphia, PA, 2000.

[38] Evin J Cramer, J E Dennis Jr, Paul D Frank, Robert Michael Lewis, and Gregory R Shubin. Problem Formulation for Multidisciplinary Optimization. *SIAM Journal on Optimization*, 4 (4):754–776, 1994. doi: 10.1137/0804044.

[39] George B Dantzig and Phillip Wolfe. Decomposition Principle for Linear Programs. *Operations Research*, 8:101–111, 1960. doi: 10.1287/opre.8.1.101.

[40] A J de Wit and F van Keulen. Numerical Comparison of Multilevel Optimization Techniques. In *Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, April 2007.

[41] A J de Wit and F van Keulen. Overview of Methods for Multi-Level and/or Multi-Disciplinary Optimization. In *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number April, Orlando, FL, April 2010.

[42] Angel-Victor DeMiguel. *Two Decomposition Algorithms for Nonconvex Optimization Problems with Global Variables*. PhD thesis, Stanford University, 2001.

[43] Angel-Victor DeMiguel and Walter Murray. An Analysis of Collaborative Optimization Methods. In *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization*, Long Beach, CA, 2000.

[44] Victor DeMiguel and Walter Murray. A Local Convergence Analysis of Bilevel Decomposition Algorithms. *Optimization and Engineering*, 7(2):99–133, June 2006. doi: 10.1007/s11081-006-6835-3.

[45] R Enblom. Two-Level Numerical Optimization of Ride Comfort in Railway Vehicles. *Journal of Rail and Rapid Transit*, 220(1):1–11, March 2006. doi: 10.1243/095440905X33279.

[46] L F P Etman, M Kokkolaras, A T Hofkamp, P Y Papalambros, and J E Rooda. Coordination Specification in Distributed Optimal Design of Multilevel Systems Using the χ Language. *Structural and Multidisciplinary Optimization*, 29:198–212, 2005. doi: 10.1007/s00158-004-0467-z.

[47] Anthony V Fiacco and Garth P McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. SIAM, 1990.

[48] Ranjan Ganguli. Survey of recent developments in rotorcraft design optimization. *Journal of Aircraft*, 41(3):493–510, 2004.

[49] C Geethaikrishnan, P M Mujumdar, K Sudhakar, and V Adimurthy. A Hybrid MDO Architecture for Launch Vehicle Conceptual Design. In *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number April, Orlando, FL, April 2010.

[50] Philip Geyer. Component-oriented Decomposition for Multidisciplinary Design Optimization in Building Design. *Advanced Engineering Informatics*, 23(1):12–31, 2009. doi: 10.1016/j.aei.2008.06.008.

[51] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. doi: 10.1137/S0036144504446096.

[52] Bryan Glaz, Peretz P Friedmann, and Li Liu. Helicopter Vibration Reduction throughout the Entire Flight Envelope Using Surrogate-Based Optimization. *Journal of the American Helicopter Society*, 54:12007—-1–15, 2009. doi: 10.4050/JAHS.54.012007.

[53] Justin Gray, Kenneth T Moore, and Bret A Naylor. OpenMDAO: An Open Source Framework for Multidisciplinary Analysis and Optimization. In *13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Fort Worth, TX, September 2010.

[54] B Grossman, Z Gurdal, G J Strauch, W M Eppard, and R T Haftka. Integrated Aerodynamic/Structural Design of a Sailplane Wing. *Journal of Aircraft*, 25(9):855–860, 1988. doi: 10.2514/3.45670.

[55] B. Grossman, R.T.Haftka, P.-J. Kao, D.M.Polen, and M.Rais-Rohani. Integrated aerodynamic-structural design of a transport wing. *Journal of Aircraft*, 27(12):1050–1056, 1990.

[56] Xiaoyu Gu, John E Renaud, Leah M Ashe, Stephen M Batill, Amrjit S Budhiraja, and Lee J Krajewski. Decision-Based Collaborative Optimization. *Journal of Mechanical Design*, 124 (1):1–13, March 2002. doi: 10.1115/1.1432991.

[57] Xiaoyu (Stacey) Gu, John E Renaud, and Charles L Penninger. Implicit Uncertainty Propagation for Robust Collaborative Optimization. *Journal of Mechanical Design*, 128(4):1001–1013, July 2006. doi: 10.1115/1.2205869.

[58] R T Haftka, J Sobieszczanski-Sobieski, and S L Padula. On Options for Interdisciplinary Analysis and Design Optimization. *Structural Optimization*, 4:65–74, 1992. doi: 10.1007/BF01759919.

[59] Raphael T Haftka. Automated Procedure for Design of Wing Structures to Satisfy Strength and Flutter Requirements. Technical Report TN D-7264, NASA Langley Research Center, Hampton, VA, 1973.

[60] Raphael T. Haftka. Optimization of flexible wing structures subject to strength and induced drag constraints. *AIAA Journal*, 14(8):1106–1977, 1977.

[61] Raphael T Haftka. Simultaneous Analysis and Design. *AIAA Journal*, 23(7):1099–1103, July 1985. doi: 10.2514/3.9043.

[62] Raphael T. Haftka and C. P. Shore. Approximate methods for combined thermal/structural design. Technical Report TP-1428, NASA, June 1979.

[63] Raphael T Haftka and Layne T Watson. Multidisciplinary Design Optimization with Quasiseparable Subsystems. *Optimization and Engineering*, 6:9–20, 2005. doi: 10.1023/B:OPTE.0000048534.58121.93.

[64] Raphael T Haftka and Layne T Watson. Decomposition Theory for Multidisciplinary Design Optimization Problems with Mixed Integer Quasiseparable Subsystems. *Optimization and Engineering*, 7(2):135–149, June 2006. doi: 10.1007/s11081-006-6836-2.

[65] Jeongwoo Han and Panos Y Papalambros. A Sequential Linear Programming Coordination Algorithm for Analytical Target Cascading. *Journal of Mechanical Design*, 132(3):021003—-1–8, March 2010. doi: 10.1115/1.4000758.

[66] Jeongwoo Han and Panos Y Papalambros. A Note on the Convergence of Analytical Target Cascading with Infinite Norms. *Journal of Mechanical Design*, 132(3):034502—-1–6, March 2010. doi: 10.1115/1.4001001.

[67] Jeongwoo Han and Panos Y Papalambros. Optimal Design of Hybrid Electric Fuel Cell Vehicles Under Uncertainty and Enterprise Considerations. *Journal Of Fuel Cell Science And Technology*, 7(2):021020—-1–9, April 2010. doi: 10.1115/1.3179762.

[68] Yuping He and John Mcphee. Multidisciplinary Optimization of Multibody Systems with Application to the Design of Rail Vehicles. *Multibody System Dynamics*, 14(2):111–135, 2005. doi: 10.1007/s11044-005-4310-0.

[69] C.-H. Huang, J Galuski, and C L Bloebaum. Multi-Objective Pareto Concurrent Subspace Optimization for Multidisciplinary Design. *AIAA Journal*, 45(8):1894–1906, August 2007. doi: 10.2514/1.19972.

[70] George Q Huang and T Ã Qu. Extending Analytical Target Cascading for Optimal Configuration of Supply Chains with Alternative Autonomous Suppliers. *International Journal of Production Economics*, 115:39–54, 2008. doi: 10.1016/j.ijpe.2008.04.008.

[71] George Q Huang, T Qu, David W L Cheung, and L Liang. Extensible Multi-Agent System for Optimal Design of Complex Systems using Analytical Target Cascading. *International Journal of Advanced Manufacturing Technology*, 30:917–926, 2006. doi: 10.1007/s00170-005-0064-3.

[72] Hong-Zhong Huang, Ye Tao, and Yu Liu. Multidisciplinary Collaborative Optimization using Fuzzy Satisfaction Degree and Fuzzy Sufficiency Degree Model. *Soft Computing*, 12:995–1005, 2008. doi: 10.1007/s00500-007-0268-6.

[73] K F Hulme and C L Bloebaum. A Simulation-Based Comparison of Multidisciplinary Design Optimization Solution Strategies using CASCADE. *Structural and Multidisciplinary Optimization*, 19:17–35, 2000. doi: 10.1007/s001580050083.

[74] Rajesh Kalavalapally, Ravi Penmetsa, and Ramana Grandhi. Multidisciplinary optimization of a lightweight torpedo structure subjected to an underwater explosion. *Finite Elements in Analysis and Design*, 43(2):103–111, December 2006. doi: 10.1016/j.finel.2006.07.005.

[75] Graeme J. Kennedy and Joaquim R. R. A. Martins. Parallel solution methods for aerostructural analysis and design optimization. In *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Forth Worth, TX, September 2010. AIAA 2010-9308.

[76] Graeme J Kennedy, Joaquim R R A Martins, and Jorn S Hansen. Aerostructural Optimization of Aircraft Structures Using Asymmetric Subspace Optimization. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimizaton Conference*, Victoria, BC, Canada, 2008. AIAA.

[77] H M Kim, M Kokkolaras, L S Louca, G J Delagrammatikas, N F Michelena, Z S Filipi, P Y Papalambros, J L Stein, and D N Assanis. Target cascading in vehicle redesign: a class VI truck study. *International Journal of Vehicle Design*, 29(3):199–225, 2002.

[78] Harrison M Kim, Wei Chen, and Margaret M Wiecek. Lagrangian Coordination for Enhancing the Convergence of Analytical Target Cascading. *AIAA Journal*, 44(10):2197–2207, October 2006. doi: 10.2514/1.15326.

[79] Hongman Kim, Scott Ragon, Grant Soremekun, Brett Malone, and Jaroslaw Sobieszczanski-Sobieski. Flexible Approximation Model Approach for Bi-Level Integrated System Synthesis. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, pages 1–11, August 2004.

[80] Hyung Min Kim. *Target Cascading in Optimal System Design.* Phd thesis, University of Michigan, 2001.

[81] Hyung Min Kim, Nestor F Michelena, Panos Y Papalambros, and Tao Jiang. Target Cascading in Optimal System Design. *Journal of Mechanical Design*, 125(3):474–480, September 2003. doi: 10.1115/1.1582501.

[82] Hyung Min Kim, D Geoff Rideout, Panos Y Papalambros, and Jeffrey L Stein. Analytical Target Cascading in Automotive Vehicle Design. *Journal of Mechanical Design*, 125(3):481–490, September 2003. doi: 10.1115/1.1586308.

[83] Srinivas Kodiyalam. Evaluation of Methods for Multidisciplinary Design Optimization ({MDO}), Part 1. \emph{{NASA} Report} CR-2000-210313, NASA, 1998.

[84] Srinivas Kodiyalam and Jaroslaw Sobieszczanski-Sobieski. Bilevel Integrated System Synthesis with Response Surfaces. *AIAA Journal*, 38(8):1479–1485, August 2000. doi: 10.2514/2.1126.

[85] Srinivas Kodiyalam and Jaroslaw Sobieszczanski-Sobieski. Multidisciplinary Design Optimization - Some Formal Methods, Framework Requirements, and Application to Vehicle Design. *International Journal of Vehicle Design*, 25:3–22, 2001. doi: 10.1504/IJVD.2001.001904.

[86] Srinivas Kodiyalam and Charles Yuan. Evaluation of Methods for Multidisciplinary Design Optimization ({MDO}), Part 2. \emph{{NASA} Report} CR-2000-210313, NASA, November 2000.

[87] Srinivas Kodiyalam, Mark Kremenetsky, and Stan Posey. Balanced HPC Infrastructure for CFD and Associated Multi-Discipline Simulations of Engineering Systems. *Journal of Aerospace Sciences and Technologies*, 61(3):434–443, 2009.

[88] M Kokkolaras, R Fellini, H M Kim, N F Michelena, and P Y Papalambros. Extension of the Target Cascading Formulation to the Design of Product Families. *Structural and Multidisciplinary Optimization*, 24:293–301, 2002. doi: 10.1007/s00158-002-0240-0.

[89] M Kokkolaras, L S Louca, G J Delagrammatikas, N F Michelena, Z S Filipi, P Y Papalambros, J L Stein, and D N Assanis. Simulation-based Optimal Design of Heavy Trucks by Model-based Decomposition: an Extensive Analytical Target Cascading Case Study. *International Journal of Heavy Vehicle Systems*, 11:403–433, 2004. doi: 10.1504/IJHVS.2004.005456.

[90] Michael Kokkolaras, Zissimous P Mourelatos, and Panos Y Papalambros. Design Optimization of Hierarchically Decomposed Multilevel Systems Under Uncertainty. *Journal of Mechanical Design*, 128(2):503–508, March 2006. doi: 10.1115/1.2168470.

[91] Ilan M Kroo. {MDO} for Large-Scale Design. In N Alexandrov and M Y Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the-Art*, pages 22–44. SIAM, 1997.

[92] Ilan M Kroo, Steve Altus, Robert Braun, Peter Gage, and Ian Sobieski. Multidisciplinary Optimization Methods for Aircraft Preliminary Design. In *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1994.

[93] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 2012. doi: 10.1007/s00158-012-0763-y. (In press).

[94] Leon S Lasdon. *Optimization Theory for Large Systems*. The Macmillan Company, 1970.

[95] Laura A Ledsinger and John R Olds. Optimized Solutions for Kistler K-1 Branching Trajectories Using Multidisciplinary Design Optimization Techniques. *Journal of Spacecraft and Rockets*, 39(3):420–429, May 2002. doi: 10.2514/2.3825.

[96] Patrick A Legresley and Juan J Alonso. Improving the Performance of Design Decomposition Methods with POD. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, August 2004.

[97] Xiang Li, Weiji Li, and Chang'an Liu. Geometric Analysis of Collaborative Optimization. *Structural and Multidisciplinary Optimization*, 35:301–313, 2008. doi: 10.1007/s00158-007-0127-1.

[98] Yanjing Li, Zhaosong Lu, and Jeremy J Michalek. Diagonal Quadratic Approximation for Parallelization of Analytical Target Cascading. *Journal of Mechanical Design*, 130(5):051402—-1–11, May 2008. doi: 10.1115/1.2838334.

[99] Zhijun Li, Michael Kokkolaras, Panos Y Papalambros, and S Jack Hu. Product and Process Tolerance Allocation in Multistation Compliant Assembly Using Analytical Target Cascading. *Journal of Mechanical Design*, 130(9):091701—-1–9, September 2008. doi: 10.1115/1.2943296.

[100] JiGuan G Lin. Analysis and Enhancement of Collaborative Optimization for Multidisciplinary Design. *AIAA Journal*, 42(2):348–360, February 2004. doi: 10.2514/1.9098.

[101] B Liu, R T Haftka, and L T Watson. Global-Local Structural Optimization Using Response Surfaces of Local Optimization Margins. *Structural and Multidisciplinary Optimization*, 27 (5):352–359, July 2004. doi: 10.1007/s00158-004-0393-0.

[102] Huibin Liu, Wei Chen, Michael Kokkolaras, Panos Y Papalambros, and Harrison M Kim. Probabilistic Analytical Target Cascading: A Moment Matching Formulation for Multilevel Optimization Under Uncertainty. *Journal of Mechanical Design*, 128(4):991–1000, July 2006. doi: 10.1115/1.2205870.

[103] E. Livne, L.A. Schmit, and P.P. Friedmann. Towards integrated multidisciplinary synthesis of actively controlled fiber composite wings. *Journal of Aircraft*, 27(12):979–992, December 1990. doi: 10.2514/3.45972.

[104] Eli Livne. Integrated aeroservoelastic optimization: Status and direction. *Journal of Aircraft*, 36(1):122–145, 1999.

[105] Charles A. Mader, Joaquim R. R. A. Martins, Juan J. Alonso, and Edwin van der Weide. ADjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA Journal*, 46(4):863–873, April 2008. doi: 10.2514/1.29123.

[106] Valerie M Manning. *Large-Scale Design of Supersonic Aircraft via Collaborative Optimization*. PhD thesis, Stanford University, 1999.

[107] Christopher Marriage. *Automatic Implementation of Multidisciplinary Design Optimization Architectures Using πMDO*. Master's thesis, University of Toronto, 2008.

[108] Christopher J Marriage and Joaquim R R A Martins. Reconfigurable Semi-Analytic Sensitivity Methods and MDO Architectures within the πMDO Framework. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimizaton Conference*, Victoria, BC, Canada, September 2008.

[109] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003. doi: 10.1145/838250.838251.

[110] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523–530, 2004. doi: 10.2514/1.11478.

[111] Joaquim R. R. A. Martins, Juan J. Alonso, and James J. Reuther. A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering*, 6(1):33–62, March 2005. doi: 10.1023/B:OPTE.0000048536.47956.62.

[112] Joaquim R R A Martins, Christopher Marriage, and Nathan Tedford. pyMDO: An Object-Oriented Framework for Multidisciplinary Design Optimization. *ACM Transactions on Mathematical Software*, 36(4):20:1–25, 2009. doi: 10.1145/1555386.1555389.

[113] C D McAllister, T W Simpson, K Hacker, K Lewis, and A Messac. Integrating Linear Physical Programming within Collaborative Optimization for Multiobjective Multidisciplinary Design Optimization. *Structural and Multidisciplinary Optimization*, 29(3):178–189, March 2005. doi: 10.1007/s00158-004-0481-1.

[114] Charles D McAllister and Timothy W Simpson. Multidisciplinary Robust Design Optimization of an Internal Combustion Engine. *Journal of Mechanical Design*, 125(1):124–130, March 2003. doi: 10.1115/1.1543978.

[115] Jeremy J Michalek and Panos Y Papalambros. An Efficient Weighting Update Method to Achieve Acceptable Consistency Deviation in Analytical Target Cascading. *Journal of Mechanical Design*, 127(March):206–214, March 2005. doi: 10.1115/1.1830046.

[116] Jeremy J Michalek and Panos Y Papalambros. BB-ATC: Analytical Target Cascading using Branch and Bound for Mixed Integer Nonlinear Programming. In *Proceedings of the ASME Design Engineering Technical Conference*, 2006.

[117] Jeremy J Michalek, Fred M Feinberg, and Panos Y Papalambros. Linking Marketing and Engineering Product Design Decisions via Analytical Target Cascading. *Journal of Product Innovation Management*, 22(1):42–62, 2005. doi: 10.1111/j.0737-6782.2005.00102.x.

[118] Nestor F Michelena, Hyungju Park, and Panos Y Papalambros. Convergence Properties of Analytical Target Cascading. *AIAA Journal*, 41(5):897–905, May 2003. doi: 10.2514/2.2025.

[119] Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer-Verlag, 2nd edition, 2006.

[120] Sharon L Padula and Ronnie E Gillian. Multidisciplinary Environments: A History of Engineering Framework Development. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Porthsmouth, VA, September 2006.

[121] Sharon L Padula, Natalia Alexandrov, and Lawrence L Green. MDO Test Suite at NASA Langley Research Center. In *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue,˜WA, 1996.

[122] S Parashar and C L Bloebaum. Multi-objective genetic algorithm concurrent subspace optimization (MOGACSSO) for Multidisciplinary design. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, May 2006.

[123] S Parashar and C L Bloebaum. Robust Multi-Objective Genetic Algorithm Concurrent Subspace Optimization (R-MOGACSSO) for Multidisciplinary Design. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, September 2006.

[124] Ruben E Perez. *A Multidisciplinary Optimization Framework for Flight Dynamics and Control Integration in Aircraft Design*. PhD thesis, University of Toronto, 2007.

[125] Ruben E Perez, Hugh H T Liu, and Kamran Behdinan. Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, number September, Albany, NY, August 2004.

[126] Ruben E Perez, Hugh H T Liu, and Kamran Behdinan. Multidisciplinary Optimization Framework for Control-Configuration Integration in Aircraft Conceptual Design. *Journal of Aircraft*, 43(6):1937–1948, November 2006. doi: 10.2514/1.22263.

[127] Daniele Peri and Emilio F. Campana. Multidisciplinary design optimization of a naval surface combatant. *Journal of Ship Research*, 47(1):1–12, 2003.

[128] Benjamin Potsaid, Yves Bellouard, and John Ting-Yung Wen. A Multidisciplinary Design and Optimization Methodology for the Adaptive Scanning Optical Microscope (ASOM). *Proceedings of the SPIE*, 6289(May 2010):62890L1–12, 2006. doi: 10.1117/12.680450.

[129] J E Renaud and G A Gabriele. Improved Coordination in Nonhierarchic System Optimization. *AIAA Journal*, 31(12):2367–2373, December 1993. doi: 10.2514/3.11938.

[130] J E Renaud and G A Gabriele. Approximation in Non-Hierarchic System Optimization. *AIAA Journal*, 32(1):198–205, 1994. doi: 10.2514/3.11967.

[131] T D Robinson, K E Willcox, M S Eldred, and R Haimes. Multifidelity Optimization for Variable Complexity Design. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2006.

[132] Brian Roth and Ilan Kroo. Enhanced Collaborative Optimization: Application to an Analytic Test Problem and Aircraft Design. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimizaton Conference*, number September, Victoria, British Columbia, Canada, September 2008.

[133] Brian D Roth. *Aircraft Family Design using Enhanced Collaborative Optimization*. phdthesis, Stanford University, 2008.

[134] Andrzej Ruszczynski. On Convergence of an Augmented Lagrangian Decomposition Method for Sparse Convex Optimization. *Mathematics of Operations Research*, 20(3):634–656, 1995. doi: 10.1287/moor.20.3.634.

[135] L A Schmit. Structural Design by Systematic Synthesis. In *2nd Conference on Electronic Computation*, pages 105–132, New York, NY, 1960. ASCE.

[136] L. A. Schmit, Jr. Structural synthesis — precursor and catalyst. recent experiences in multidisciplinary analysis and optimization. Technical Report CP-2337, NASA, 1984.

[137] Lucien A Schmit and William A Thornton. Synthesis of an Airfoil at Supersonic Mach Number. Technical Report CR 144, NASA, January 1965.

[138] L A Schmit Jr. and R K Ramanathan. Multilevel approach to minimum weight design including buckling constraints. *AIAA Journal*, 16(2):97–104, February 1978. doi: 10.2514/3.60867.

[139] R S Sellar, S M Batill, and J E Renaud. Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design. In *34th AIAA Aerospace Sciences and Meeting Exhibit*, 1996.

[140] M K Shin, B S Kang, and G J Park. Application of the Multidisciplinary Design Optimization Algorithm to the Design of a Belt-Integrated Seat while Considering Crashworthiness. *Journal of Automobile Engineering*, 219(11):1281–1292, November 2005. doi: 10.1243/095440705X34928.

[141] Moon-Kyun Shin and Gyung-Jin Park. Multidisciplinary Design Optimization based on Independent Subspaces. *International Journal for Numerical Methods in Engineering*, 64: 599–617, 2005. doi: 10.1002/nme.1380.

[142] Ian P Sobieski and Ilan M Kroo. Collaborative Optimization Using Response Surface Estimation. *AIAA Journal*, 38(10):1931–1938, 2000. doi: 10.2514/2.847.

[143] J Sobieszczanski-Sobieski and R T Haftka. Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments. *Structural Optimization*, 14(1):1–23, 1997. doi: 10.1007/BF01197554.

[144] J. Sobieszczanski-Sobieski and R. T. Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization*, 14(1):1–23, 1997.

[145] Jaroslaw Sobieszczanski-Sobieski. Optimization by Decomposition: a Step from Hierarchic to Non-Hierarchic Systems. Technical Report September, NASA Langley Research Center, Hampton, VA, 1988.

[146] Jaroslaw Sobieszczanski-Sobieski. Sensitivity of Complex, Internally Coupled Systems. *AIAA Journal*, 28(1):153–160, April 1990. doi: 10.2514/3.10366.

[147] Jaroslaw Sobieszczanski-Sobieski. Integrated System-of-Systems Synthesis. *AIAA Journal*, 46(5):1072–1080, May 2008. doi: 10.2514/1.27953.

[148] Jaroslaw Sobieszczanski-Sobieski. Sensitivity of complex, internally coupled systems. *AIAA Journal*, 28(1):153–160, 1990.

[149] Jaroslaw Sobieszczanski-Sobieski, Jeremy S Agte, and Robert R Sandusky Jr. Bilevel Integrated System Synthesis. *AIAA Journal*, 38(1):164–172, January 2000. doi: 10.1.1.35.4601.

[150] Jaroslaw Sobieszczanski-Sobieski, Troy D Altus, Matthew Phillips, and Robert R Sandusky Jr. Bilevel Integrated System Synthesis for Concurrent and Distributed Processing. *AIAA Journal*, 41(10):1996–2003, October 2003. doi: 10.2514/2.1889.

[151] D V Steward. The Design Structure Matrix: A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management*, 28:71–74, 1981.

[152] Jun Tajima, Fujio Momiyama, and Naohiro Yuhara. A New Solution for Two-Bag Air Suspension System with Leaf Spring for Heavy-Duty Vehicle. *Vehicle System Dynamics*, 44(2): 107–138, February 2006. doi: 10.1080/00423110500385907.

[153] R V Tappeta and J E Renaud. Multiobjective Collaborative Optimization. *Journal of Mechanical Design*, 119:403–411, 1997. doi: 10.1115/1.2826362.

[154] R V Tappeta, S Nagendra, and J E Renaud. A Multidisciplinary Design Optimization Approach for High Temperature Aircraft Engine Components. *Structural Optimization*, 18(2-3): 134–145, October 1999. doi: 10.1007/BF01195988.

[155] Nathan P Tedford and Joaquim R R A Martins. Benchmarking Multidisciplinary Design Optimization Algorithms. *Optimization and Engineering*, 11:159–183, 2010. doi: 10.1007/s11081-009-9082-6.

[156] S Tosserams, L F P Etman, P Y Papalambros, and J E Rooda. An Augmented {Lagrangian} Relaxation for Analytical Target Cascading using the Alternating Direction Method of Multipliers. *Structural and Multidisciplinary Optimization*, 31(3):176–189, March 2006. doi: 10.1007/s00158-005-0579-0.

[157] S Tosserams, L F P Etman, and J E Rooda. An Augmented {Lagrangian} Decomposition Method for Quasiseparable Problems in {MDO}. *Structural and Multidisciplinary Optimization*, 34:211–227, 2007. doi: 10.1007/s00158-006-0077-z.

[158] S Tosserams, L F P Etman, and J E Rooda. Augmented {Lagrangian} Coordination for Distributed Optimal Design in {MDO}. *International Journal for Numerical Methods in Engineering*, 73:1885–1910, 2008. doi: 10.1002/nme.2158.

[159] S Tosserams, L F P Etman, and J E Rooda. A Classification of Methods for Distributed System Optimization based on Formulation Structure. *Structural and Multidisciplinary Optimization*, 39(5):503–517, 2009. doi: 10.1007/s00158-008-0347-z.

[160] S Tosserams, L F P Etman, and J E Rooda. Block-Separable Linking Constraints in Augmented Lagrangian Coordination in {MDO}. *Structural and Multidisciplinary Optimization*, 37(5):521–527, 2009. doi: 10.1007/s00158-008-0244-5.

[161] S Tosserams, L F P Etman, and J E Rooda. Multi–Modality in Augmented Lagrangian Coordination for Distributed Optimal Design. *Structural and Multidisciplinary Optimization*, 40:329–352, 2010. doi: 10.1007/s00158-009-0371-7.

[162] S Tosserams, L F P Etman, and J E Rooda. A Micro-accelerometer {MDO} Benchmark Problem. *Structural and Multidisciplinary Optimization*, 41(2):255–275, 2010. doi: 10.1007/s00158-009-0422-0.

[163] S Tosserams, M Kokkolaras, L F P Etman, and J E Rooda. A Nonhierarchical Formulation of Analytical Target Cascading. *Journal of Mechanical Design*, 132(5):051002—-1–13, May 2010. doi: 10.1115/1.4001346.

[164] G N Vanderplaats. An Efficient Feasible Directions Algorithm for Design Synthesis. *AIAA Journal*, 22(11):1633–1640, November 1984. doi: 10.2514/3.8829.

[165] Stephen J Wright. *Primal-Dual Interior Point Methods*. SIAM, 1997.

[166] Brett A Wujek, John E Renaud, Stephen M Batill, and Jay B Brockman. Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment. *Concurrent Engineering: Research and Applications*, 4(4):361–377, 1996. doi: 10.1177/1063293X9600400405.

[167] S I Yi, J K Shin, and G J Park. Comparison of MDO Methods with Mathematical Examples. *Structural and Multidisciplinary Optimization*, 39:391–402, 2008. doi: 10.1007/s00158-007-0150-2.

[168] Parviz M Zadeh and Vassili V Toropov. Multi-Fidelity Multidisciplinary Design Optimization based on Collaborative Optimization Framework. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, number September, Atlanta, GA, September 2002.

[169] Parviz M Zadeh, Vassili V Toropov, and Alastair S Wood. Metamodel-based Collaborative Optimization Framework. *Structural and Multidisciplinary Optimization*, 38:103–115, 2009. doi: 10.1007/s00158-008-0286-8.

[170] Ke-Shi Zhang, Zhong-Hua Han, Wei-Ji Li, and Wen-Ping Song. Bilevel Adaptive Weighted Sum Method for Multidisciplinary Multi-Objective Optimization. *AIAA Journal*, 46(10): 2611–2622, October 2008. doi: 10.2514/1.36853.