Multidisciplinary Design Optimization: A Survey of Architectures

Joaquim R. R. A. Martins*
University of Michigan, Ann Arbor, MI
Andrew B. Lambe†
University of Toronto, Toronto, ON, Canada

Multidisciplinary design optimization (MDO) is a field of research that studies the application of numerical optimization techniques to the design of engineering systems involving multiple disciplines or components. Since the inception of MDO, various methods (architectures) have been developed and applied to solve MDO problems. This paper provides a survey of all the architectures that have been presented in the literature so far. All architectures are explained in detail using a unified description that includes optimization problem statements, diagrams, and detailed algorithms. The diagrams show both data and process flow through the multidisciplinary system and computational elements, which facilitates the understanding of the various architectures, and how they relate to each other. A classification of the MDO architectures based on their problem formulations and decomposition strategies is also provided, and the benefits and drawbacks of the architectures are discussed from both a theoretical and experimental perspective. For each architecture, several applications to the solution of engineering design problems are cited. The result is a comprehensive but straightforward introduction to MDO for non-specialists, and a reference detailing all current MDO architectures for specialists.

I. Introduction

Multidisciplinary design optimization (MDO) is a field of engineering that focuses on the use of numerical optimization for the design of systems that involve a number of disciplines or subsystems. The main motivation for using MDO is that the performance of a multidisciplinary system is driven not only by the performance of the individual disciplines but also by their interactions. Considering these interactions in an optimization problem generally 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 [1, 2, 3, 4] and Haftka [5, 6, 7, 8], 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 [9, 10, 11, 12, 13, 14, 15, 16]. Since then, the application of MDO has been extended to complete aircraft [17, 18, 19, 20, 21] and a wide range of other engineering systems, such as bridges [22], buildings [23, 24], railway cars [25, 26], microscopes [27], automobiles [28, 29], ships [30, 31], propellers [32, 33], rotorcraft [34, 35], wind turbines [36, 37, 38], and spacecraft [39, 40].

One of the most important considerations when implementing MDO is how to organize the discipline analysis models, approximation models (if any), and optimization software in concert with the problem formulation so that an optimal design is achieved. (In this work, we mean "optimal" in the local sense since it is in general difficult to assess global optimality.) Such a combination of problem formulation and organizational strategy is referred to as an

*Associate Professor, Department of Aerospace Engineering, AIAA Associate Fellow, jrram@umich.edu
†PhD Candidate, Institute for Aerospace Studies, AIAA Student Member, lambe@utias.utoronto.ca

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 a monolithic approach, a single optimization problem is solved. In a distributed approach, the same 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—the choice of the architecture has a significant influence on both the solution time and the final design. For example, using a global optimization algorithm rather than a gradient-based algorithm may lead to a better final design because the gradient-based optimizer may converge to a local minimum early in the design process. However, if gradients can be computed efficiently, the computational cost of the gradient-based optimization may be far less than that of the global optimization because the discipline analyses do not need to be run as many times [41]. If the calculations required by a given architecture are easy to run in parallel and if parallel computing facilities are available, a distributed architecture may be preferred over a monolithic architecture despite the increased computational expense. Human and computational organization can also play a role. In practice, careful consideration of the human and computing environment, the available algorithms, and the design problem at hand is necessary to decide on the most appropriate MDO architecture.

In this survey, we are primarily focused on methods for solving MDO problems with a single objective function and continuous design variables. It is assumed that the optimality of a given design corresponds to the satisfaction of the Karush–Kuhn–Tucker (KKT) optimality conditions (see, for example, Nocedal and Wright [42] for a comprehensive mathematical definition). These conditions are necessary for local optimality, so it is possible for different architectures to obtain different (yet equally valid) local optima. The KKT conditions require the availability of function gradients, so we assume that the objective and constraint functions are differentiable. However, various architectures have been developed specifically for multiobjective problems [43, 44, 45, 28] or problems with discrete variables [46, 47]. Other architectures determine optimality without using gradients [48, 49] or use optimality concepts from game theory [50, 51, 52]. Whenever possible, we comment on the connections with the work that goes beyond the assumptions of this survey.

In the MDO literature, several terms are used to describe what we call "architecture": "method" [17, 53, 54, 55, 56], "methodology" [57, 58, 59], "problem formulation" [60, 61, 62, 63], "strategy" [64, 65], "procedure" [66, 67], and "algorithm" [68, 69, 70, 71] have all been used. Some authors use a variety of terms and occasionally use them interchangeably in the same paper. Our preference for the term "architecture" [72, 73, 74, 75] comes from the fact that the relationship between the problem formulation and the solution algorithm is not one-to-one. For example, replacing a particular discipline analysis with a surrogate model or reordering the discipline analyses does not affect the problem formulation but strongly affects the solution algorithm.

There have been a number of surveys of MDO over the last two decades. Haftka et al. [76] were among the first to review the MDO architectures known at the time. Cramer et al. [60] formalized the monolithic architectures and detailed the required gradient computation methods. Balling and Sobieski [77] identified a number of possible monolithic approaches and estimated their computational cost. In a collection of articles entitled "Multidisciplinary Design Optimization: State of the Art" edited by Alexandrov and Hussaini [78], Kroo [73] provided a comprehensive overview of MDO, including a description of both monolithic and distributed architectures. In the same volume, Alexandrov [79] discussed the convergence properties of certain partitioning strategies for distributed architectures, and Balling [80] focused on partitioning as a way to provide disciplinary autonomy. In the same year, Sobieski and Haftka [66] 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 have been developed, and various successful applications of MDO have taken place in industry [81, 82, 56, 83]. A recent paper by Tosserams et al. [84] 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 the existing architectures that

compares the features, merits, and performance of each.

The purpose of this paper 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 because it is able to highlight similarities between architectures that were developed independently.

For readers who want a quick overview, it is possible to skip most of the text and still learn the essence of the various MDO architectures by: 1) Consulting the mathematical notation in Table 1, 2) Studying the problem formulation and pseudocode for each architecture and 3) Consulting the MDO architecture classification diagram in Fig. 7.

We have organized this paper as follows. In Sec. II we present the unified notation and diagrams for describing MDO architectures. In Sec. III we define the general MDO problem and describe the three basic monolithic architectures and their derivation from a common problem statement. In Sec. IV we focus on the distributed architectures; we discuss the motivation for using these methods and their mathematical derivation. We also describe a new classification of distributed architectures by drawing parallels with the monolithic architectures, and we then explain the distributed MDO architectures in detail, discussing their capabilities. In Sec. V 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. Finally, we summarize our conclusions in Sec. VI and comment on possible future research in the field.

II. Unified Description of MDO Architectures

A. Terminology and Mathematical Notation

Before introducing the mathematical background, we introduce the notation that we use throughout this paper. This notation allows us to compare the various problem formulations within the architectures and to identify how similar features of the general MDO problem are handled in each case. The notation is listed in Table 1. This is not a comprehensive list; additional notation specific to particular architectures will be introduced as necessary. We also take this opportunity to clarify many of the terms we use that are specific to the field of MDO.

Table 1. Mathematical notation for MDO problem formulations

Symbol	Definition
x	Vector of design variables
y	Vector of coupling variables (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 (discipline analysis constraints)
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
$\tilde{()}$	Approximations of a given function or vector of functions
$\hat{()}$	Independent copies of variables distributed to other disciplines

A *design variable* is a quantity in the MDO problem that is always under the explicit control of an optimizer. In

traditional engineering design, the values of these variables are selected explicitly by the designer or design team. Design variables may be *local*, i.e., pertain to a single discipline, or they may be *shared* by multiple disciplines. We denote the vector of design variables local to discipline i by x_i and the 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 when 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, the static equilibrium equations in structural mechanics, or the equations of motion in a control simulation—to 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, the structural analysis does not require the states for the entire flow field. Instead, only the surface aerodynamic loads are required. 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 [60]. Similarly, a transformation may be needed to convert input coupling variables into a usable format within each discipline [60]. 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 to run independently and in parallel. These copies, which function as design variables in the problem formulation, are sometimes called *target variables*. We denote the coupling variable copies by $\hat{y}$. For example, the copy of the coupling variables produced by discipline i is denoted $\hat{y}_i$. These variables are independent of the corresponding original variables and are used as part of the input to disciplines that are coupled to discipline i through y_i. To preserve consistency between the coupling variable inputs and outputs at the optimal solution, we define a set of *consistency constraints*, $c_i^c = \hat{y}_i - y_i$, which we add to the optimization problem formulation.

B. 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 merely present the problem formulation and leave the readers to work out the implementation. 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 exposition, each of the architectures is presented with a new diagram that we call the *extended design structure matrix*, or XDSM [85]. As the name suggests, the XDSM was based on the design structure matrix (DSM) [86, 87], a common diagram in systems engineering that is used to visualize the interconnections among components of a complex system. The traditional DSM shows components and connections between components, but the meaning of the connections is left ambiguous. To represent MDO architectures, we need two types of connections: data dependency and process flow. This need motivated the development of XDSM, which simultaneously com-

municates data dependency and process flow between computational components of MDO architectures on a single diagram. We present a brief overview of the XDSM; for further details see Lambe and Martins [85].

We explain the basics of the XDSM using two simple examples. The first example is shown in Fig. 1, and represents a Gauss–Seidel multidisciplinary analysis (MDA) procedure for three disciplines. The pseudocode for this procedure is listed in Algorithm 1. As with the traditional DSM, the components are laid out along the diagonal. The components in this case consist of the discipline analyses and a special component, known as a *driver*, that controls the iteration and is represented by a rounded rectangle. The function of the components is to process data. The data flow is shown as thick gray lines. The components take data inputs from the vertical direction and output data in the horizontal direction. Thus, the connections above the diagonal flow from left to right and top to bottom, and the connections below the diagonal flow from right to left and bottom to top. The off-diagonal nodes in the shape of parallelograms are used to label the data. Using this convention, it is easy to identify the inputs of a given component by scanning the column above and below the component, while the outputs can be identified by scanning the row. External inputs and outputs are placed on the outer edges of the diagram, in the top row and leftmost column, respectively. In Fig. 1, the external inputs are the design variables and an initial guess of the system coupling variables. Each discipline analysis computes its own set of coupling variables that 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.

The thin black lines show the process flow. The direction of these lines follows the convention for the data-flow lines. In addition, a numbering system is used to show the order in which the components are executed. These numbers are presented inside each component in the diagram followed by a colon and the component name. As the algorithm executes, every time a number is reached, the corresponding component performs a relevant computation. Multiple numbers indicate that the component is called several times within the algorithm. The algorithm starts at component zero and proceeds in numerical order, following the process lines. Loops are denoted by $j \rightarrow k$ for $k < j$, indicating that the algorithm must return to step k until some condition required by the driver is satisfied. The data nodes are also labeled with numbers to indicate when the input data is retrieved.

Algorithm 1 Block Gauss–Seidel multidisciplinary analysis for three disciplines

Input: Design variables x

Output: Coupling variables y

0: Initiate MDA iteration loop

repeat

1: Evaluate Analysis 1 and update $y_1(y_2, y_3)$

2: Evaluate Analysis 2 and update $y_2(y_1, y_3)$

3: Evaluate Analysis 3 and update $y_3(y_1, y_2)$

until $4 \rightarrow 1$: MDA has converged

The second XDSM example, illustrated in Fig. 2, is the solution process for an analytic optimization problem using gradient-based optimization. The problem has a single objective and a vector of constraints. Figure 2 shows separate components to compute the objective, the constraints, and their gradients, as well as a driver to control the iteration. We assume that the gradients can be computed without knowledge of the objective and constraint function values. Notice that multiple components are evaluated at step one of the algorithm; this numbering denotes parallel execution. In some cases, it may be advisable to combine components, e.g., the objective and the constraints, to reflect underlying problem structures. In the following sections, we have done so to simplify the presentation; we will mention other simplifications as we proceed.

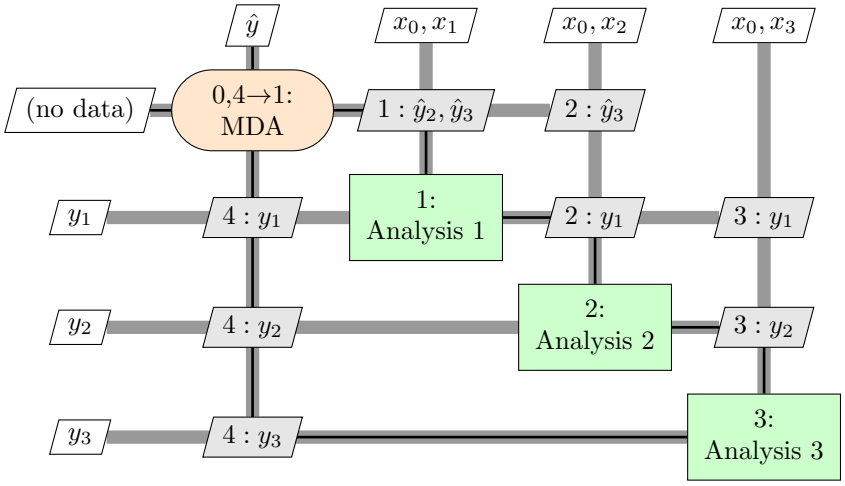

Figure 1. Block Gauss–Seidel multidisciplinary analysis (MDA) process to solve a three-discipline coupled system.

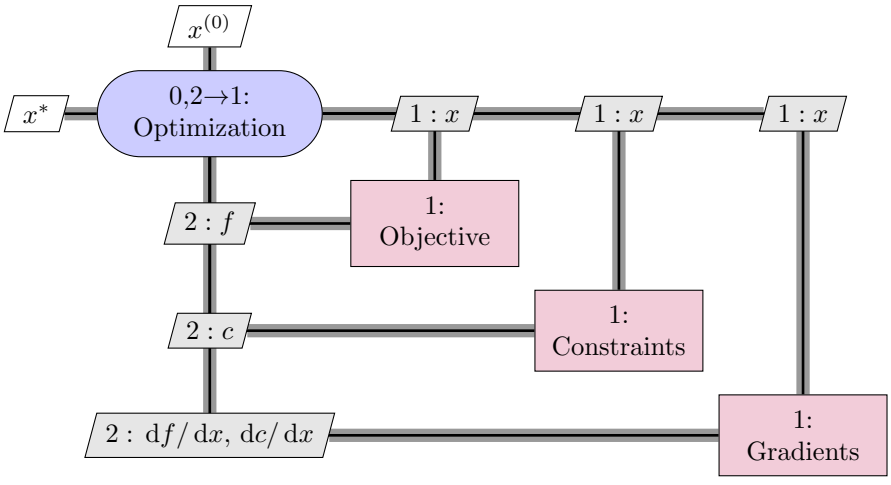

Figure 2. Gradient-based optimization procedure.

III. Monolithic Architectures

A. Introduction

If we ignore the discipline boundaries, an MDO problem is nothing more than a standard constrained nonlinear programming problem: we must find the values of the design variables that maximize or minimize a particular objective function, subject to the constraints. The choice of the objective, the 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 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 manage the coupling of the system under consideration. Like the disciplines they model, the discipline analyses are mutually interdependent: one analysis requires the outputs of other analyses as input. Furthermore, the objective and constraint functions, in general, depend on both the design variables and the 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 generally leads to a more accurate representation of the behavior of the system. MDO architectures provide a consistent, formal setting for managing this interdependence in the design process [73].

The architectures presented in this section are referred to as *monolithic* architectures. Each architecture solves the MDO problem by casting it as a single optimization problem; different strategies are used to achieve multidisciplinary feasibility. Architectures that decompose the optimization problem into smaller problems, i.e., *distributed* architectures, are presented in Sec. IV.

B. The All-at-Once (AAO) Problem Statement

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

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

which is known as the "all-at-once" (AAO) problem. This form of the design optimization problem includes all coupling variables, coupling variable copies, state variables, consistency constraints, and residuals of the governing equations directly in the problem statement.

For the design variables, we concatenate the discipline variable groups as follows: $x = [x_0^T, x_1^T, \ldots, x_N^T]^T$. The same is done for the coupling variables, $y = [y_0^T, y_1^T, \ldots, y_N^T]^T$, as well as the state variables. The objective functions f_i, represent local objective functions, i.e., objective functions that depend only on shared design variables and other variables that are local with respect to only one discipline (discipline i). We omit the sum of local objective functions except when necessary to highlight situations where an architecture exploits this term.

For the design constraints, $c = [c_0^T, c_1^T, \ldots, c_N^T]^T$, we adopt the convention of using only "greater than or equal to" inequalities. There is no loss of generality with this convention, since the sign can be switched for "less than or equal to" inequalities, and equalities can be stated as pairs of inequalities with opposing signs.

Figure 3 shows the XDSM for the AAO problem. To keep the diagrams compact, we adopt two conventions. First,

unlike Fig. 2, we do not explicitly show the gradient computation steps unless they are intrinsic to the architecture. Second, any block referring to discipline i represents a repeated pattern for every discipline. Thus, in Fig. 3 a residual block exists for every discipline, and each block can be executed in parallel. As an added visual cue, the "Residual i" component is displayed as a stack of similar components.

There is a conflict with the literature when it comes to the labeling of Problem (1). What most authors refer to as AAO, following the lead of Cramer et al. [60], others label as the simultaneous analysis and design (SAND) [88, 77] problem. Our AAO problem is most like what Cramer et al. refer to as simply "the most general formulation" [60]. Herein, we classify formulation (1) as the AAO problem, because it includes all the design, state, and input and output coupling variables in the problem, so the optimizer is responsible for all variables at once. The SAND architecture has a different problem statement and is presented in Sec. C.

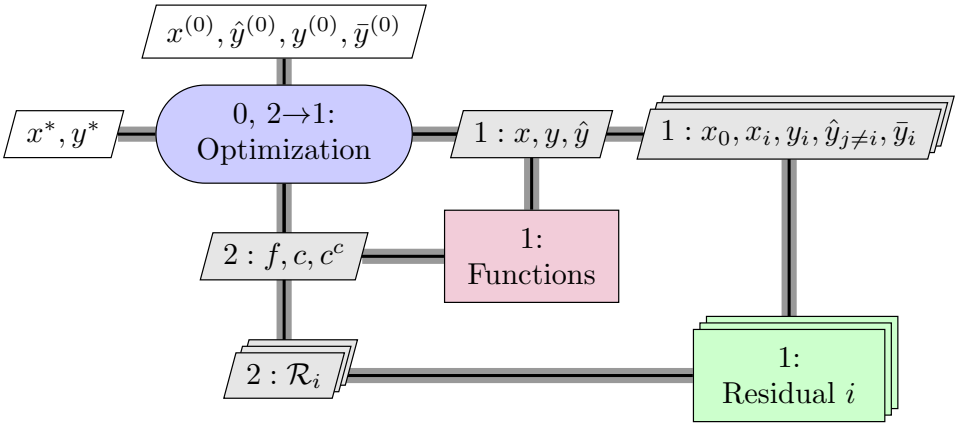

Figure 3. XDSM for the AAO problem.

The AAO problem is never solved in this form in practice because the consistency constraints, which are linear in this formulation, can be easily eliminated. 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 (1) results in the problem solved by the SAND architecture. However, we have presented the AAO problem first because it functions as a common starting point for deriving both the SAND problem and the individual discipline feasible (IDF) problem and, subsequently, all other equivalent MDO problems.

Depending on which equality constraint groups are eliminated from Problem (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 [89, 88, 60, 77]. In the next three subsections, we describe how each architecture is derived and the relative advantages and disadvantages of each. We emphasize that, in spite of the elements added or removed by each architecture, *we are always solving the same MDO problem*. Furthermore, each problem has *the same set of optimal solutions*.

C. Simultaneous Analysis and Design (SAND)

The most obvious simplification of Problem (1) is to eliminate the consistency constraints, $c_i^c = \hat{y}_i - y_i = 0$, by introducing a single group of coupling variables to replace the separate target and response groups. This simplification

yields the SAND architecture [88], which solves the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\right) \\ \text{with respect to} \quad & x, y, \bar{y} \\ \text{subject to} \quad & c_0\left(x, y\right) \geq 0 \\ & c_i\left(x_0, x_i, y_i\right) \geq 0 \qquad \text{for} \quad i = 1, \ldots, N \\ & \mathcal{R}_i\left(x_0, x_i, y, \bar{y}_i\right) = 0 \quad \text{for} \quad i = 1, \ldots, N. \end{aligned} \tag{2}$$

The XDSM for SAND is shown in Fig. 4. Cramer et al. [60] 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.

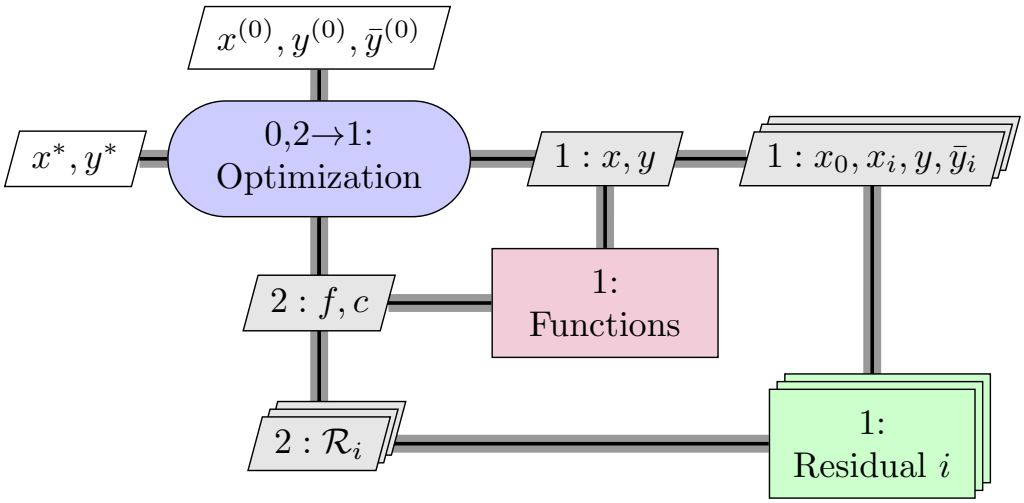

Figure 4. Diagram for the SAND architecture.

Several features of the SAND architecture are noteworthy. Because we do not need to solve any discipline analysis explicitly or exactly at each iteration, the optimization problem can potentially be solved quickly by letting the optimizer explore regions that are infeasible with respect to the analysis constraints, $\mathcal{R}_i$. The SAND methodology is not restricted to multidisciplinary systems and can be used in single-discipline optimization as well. In that case, we need to define only a single group of design constraints, c. If the discipline 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. [90] for an overview of this field.)

Two major issues are present in the SAND architecture. First, the problem formulation still requires all state variables and discipline analysis equations, so problem size and potential premature termination of the optimizer at an infeasible design can be issues in practice. Second, and more important, the fact that the discipline analysis equations are treated explicitly as constraints means that the residual values—and possibly their derivatives—need to be available to the optimizer. In other words, rather than computing coupling variables y_i and state variables $\bar{y}_i$, each discipline i accepts predetermined values of y_i and $\bar{y}_i$ and returns analysis equation residuals $\mathcal{R}_i$. (In our notation, $\mathcal{R}_i$ includes the transformation between y and $\bar{y}$ for disciplines coupled to discipline i.) In engineering design, software for discipline analysis often operates in a "black-box" fashion, directly computing the coupling variables while hiding the discipline analysis residuals and state variables. Even if the software can be modified to return the residuals, the cost and effort

required may be excessive. Therefore, most practical MDO problems require an architecture that can take advantage of existing discipline analysis software. The following two monolithic architectures address this concern.

D. Individual Discipline Feasible (IDF)

By eliminating the discipline analysis constraints $\mathcal{R}_i(x_0, x_i, y_i, \hat{y}_{j\neq i}, \bar{y}_i) = 0$ from Problem (1), we obtain the IDF architecture [60]. 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 variable copies. The IDF architecture is also known as distributed analysis optimization [61] and optimizer-based decomposition [73]. The optimization problem for the IDF architecture is

$$\begin{aligned}
\text{minimize} \quad & f_0\left(x, y\left(x, \hat{y}\right)\right) \\
\text{with respect to} \quad & x, \hat{y} \\
\text{subject to} \quad & c_0\left(x, y\left(x, \hat{y}\right)\right) \geq 0 \\
& c_i\left(x_0, x_i, y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right)\right) \geq 0 \quad \text{for} \quad i = 1, \ldots, N \\
& c_i^c = \hat{y}_i - y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right) = 0 \quad \text{for} \quad i = 1, \ldots, N.
\end{aligned} \tag{3}$$

The most important consequence of this reformulation is the removal of all the state variables and discipline analysis

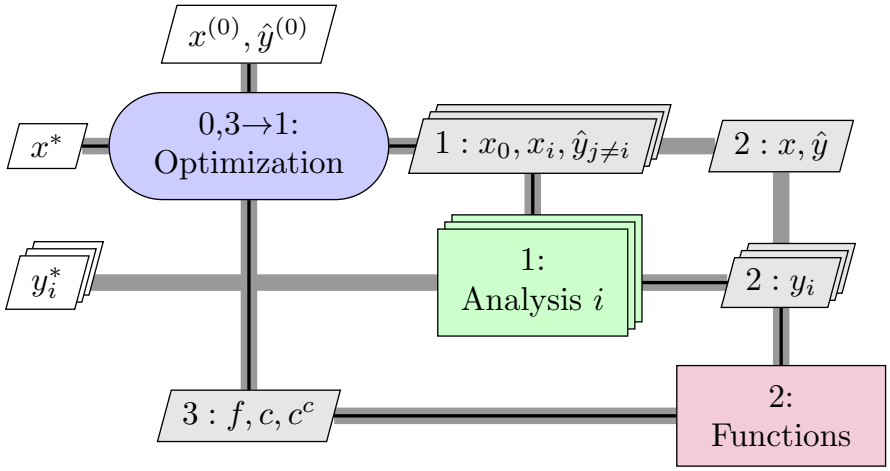

Figure 5. Diagram of the IDF architecture.

equations from the problem statement. All the coupling variables are now implicit functions of design variables and coupling variable copies as a result of the discipline analysis equations being solved exactly at each iteration.

The XDSM for IDF is shown in Fig. 5. This architecture enables the discipline analyses to be performed in parallel, since the coupling between the disciplines is resolved by the coupling variable copies, $\hat{y}$, and consistency constraints, c^c. Within the optimization iteration, specialized software for the discipline analyses can now be used to return the coupling variable values to the objective and constraint function calculations. The net effect is that the IDF problem is 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 [90].

However, the size of the IDF problem can still be an issue. If the number of coupling variables is large, the resulting optimization problem might be too large to solve efficiently. This can be mitigated to some extent by careful selection of the discipline variable partitions or aggregation of the coupling variables to reduce the information transfer between

disciplines.

If gradient-based optimization software is used to solve the IDF problem, the gradient computation can become an issue. When the discipline analyses are expensive, evaluating the objective and constraint function gradients becomes costly. This is because the gradients themselves must be discipline-feasible, i.e., the changes in the design variables cannot cause the output coupling variables to violate the discipline analysis equations to the first order.

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 discipline feasibility, it is costly and unreliable. If the discipline analysis code allows for the use of complex numbers, the complex-step method [91, 92] is an alternative approach that gives machine-precision derivative estimates. If the analysis codes require a particularly long time to evaluate, automatic differentiation or analytic derivative calculations (direct or adjoint methods) can be used to avoid multiple discipline analysis evaluations [93, 94, 95, 96, 97]. While the development time for these methods can be long, the reward is accurate derivative estimates and massive reductions in the computational cost, especially for design optimization based on high-fidelity models [98, 99, 100]. We refer the reader to the work of Martins and Hwang [101] for a detailed discussion of the options available for computing derivatives in MDO problems.

E. Multidisciplinary Feasible (MDF)

If both the analysis and consistency constraints are removed from Problem (1), we obtain the MDF architecture [60]. This architecture has also been referred to in the literature as fully integrated optimization [61] and nested analysis and design [77]. The resulting optimization problem is

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\left(x, y\right)\right) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & c_0\left(x, y\left(x, y\right)\right) \geq 0 \\ & c_i\left(x_0, x_i, y_i\left(x_0, x_i, y_{j\neq i}\right)\right) \geq 0 \quad \text{for} \quad i = 1, \ldots, N. \end{aligned} \tag{4}$$

The MDF architecture XDSM for three disciplines is shown in Fig. 6. Typically, a fixed point iteration, such as the block Gauss–Seidel iteration shown in Fig. 6, is used to converge the MDA, where each discipline is solved in turn. This approach usually exhibits slow convergence rates. Re-ordering the sequence of disciplines can improve the convergence rate of Gauss–Seidel [102], but even better convergence rates can be achieved through the use of Newton-based methods [103]. Because of 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, objective function, and design constraints are under the direct control of the optimizer. Another benefit is that MDF returns a system design that always satisfies the consistency constraints, even if the optimization process is terminated early. This is advantageous in an engineering-design context if time is limited and our concern is to find an improved design that need not be optimal in the strict mathematical sense. Note, however, that design constraint satisfaction is not guaranteed if the optimization is terminated early; this depends on whether or not the optimization algorithm maintains a feasible design point. In particular, methods of feasible directions [104] require and maintain a feasible design point while many robust sequential quadratic programming [105] and interior point methods [106] 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 optimization iteration. Instead of simply running each individual

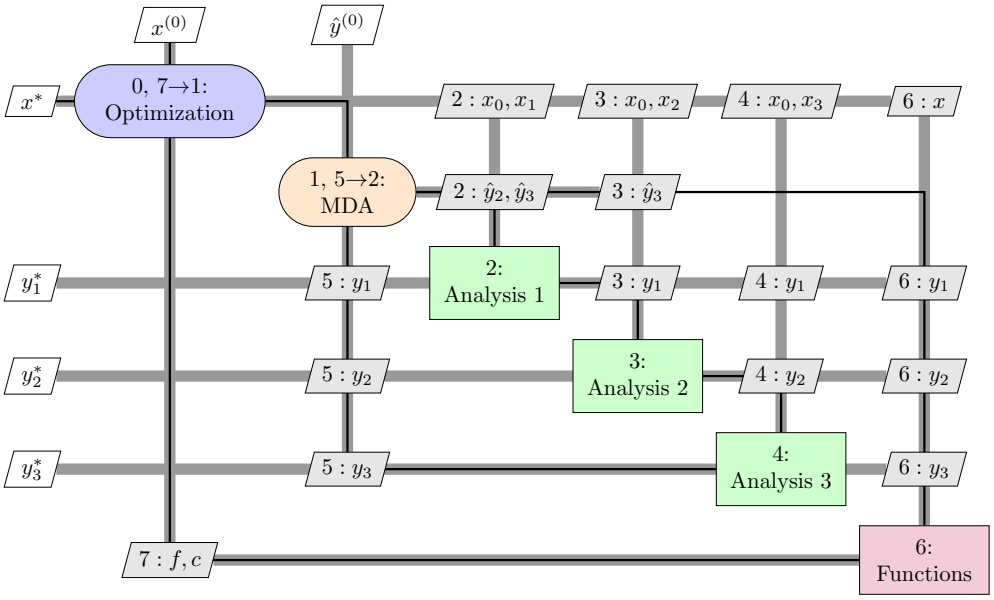

Figure 6. Diagram for the MDF architecture with a Gauss–Seidel multidisciplinary analysis.

discipline analysis once per iteration, as we do in IDF, we need to run every analysis multiple times until a consistent set of coupling variables is found. This task requires its own specialized iterative procedure outside the optimization. Developing an MDA procedure can be time-consuming.

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 in MDF must be feasible with respect to all disciplines. Fortunately, research in the sensitivity of coupled systems is fairly mature, and there are semi-analytic methods that drastically reduce the cost of this step by eliminating finite differencing over the full MDA [107, 98, 99, 101, 108]. There has also been some work on automating the implementation of these coupled sensitivity methods [109]. The required partial derivatives can be obtained using any of the methods described in Sec. D for the individual disciplines in IDF.

IV. Distributed Architectures

A. 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 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 for the development of 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 structure [110].

To better understand decomposition, consider the following problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{with respect to} \quad & x_1, \ldots, x_N \\ \text{subject to} \quad & c_0(x_1, \ldots, x_N) \geq 0 \\ & c_1(x_1) \geq 0, \ldots, c_N(x_N) \geq 0. \end{aligned} \tag{5}$$

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 depends 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 is referred to as a problem with *complicating constraints* [111]; if c_0 did not exist, we could simply decompose this optimization problem into N independent problems and solve them in parallel.

Another possibility is that a problem may include shared design variables and a separable objective function, with no complicating constraints, i.e.,

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_0, x_i) \\ \text{with respect to} \quad & x_0, x_1, \ldots, x_N \\ \text{subject to} \quad & c_1(x_0, x_1) \geq 0, \ldots, c_N(x_0, x_N) \geq 0. \end{aligned} \tag{6}$$

This is referred to as a problem with *complicating variables* [111]. 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 [112] and Benders decomposition [113] for Problems (5) and (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 by only one constraint at a time, so decomposition is the only way to exploit the special structure. 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 in the problem. Nonlinear decomposition algorithms have also been developed, but, to the best of our knowledge, no performance comparisons have been made between decomposition algorithms and Newton-like algorithms that employ sparse matrix factorization. Intuition suggests that the latter should be faster since Newton methods can exploit second-order problem information. Thus, while decomposition methods do exist for nonlinear problems, the 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 it among specific engineering groups. These groups may be geographically distributed and may communicate infrequently. Furthermore, these groups typically like to retain control of their own design procedures and make use of in-house expertise, rather than simply passing on the discipline-analysis results to a central design authority [73]. Decomposition through distributed architectures allows individual design groups to work in isolation, controlling their own sets of design variables, while periodically receiving updated information from other groups to improve their aspect of the overall design. This approach conforms more closely to current industrial design practice than does the approach of the monolithic architectures.

In an industrial setting, the notion of a "discipline" in an MDO problem can take many forms. Traditionally, disciplines have been defined in terms of knowledge areas, e.g., aircraft design disciplines include aerodynamics, structures, propulsion, and control. This definition conforms well with many existing analysis codes. In principle, however, a discipline can constitute any suitable partition of an MDO problem. For example, if the structural design is broken down by component, (e.g., wings, tail, and fuselage), the design of each component could also be considered a discipline. Therefore, an industrial-scale MDO problem could contain hundreds of disciplines, depending on the company architecture. How the disciplines are arranged within a distributed MDO architecture is up to the company, but there is some literature on choosing discipline partitions to reduce the coupling between distributed groups [114, 115, 116].

The structure of discipline 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 [117]. In instances where some analyses or optimizations are much more expensive than others, such as multifidelity optimization [118, 119], the performance suffers because the processors performing the inexpensive analyses and optimizations experience long periods of inactivity while waiting for updates from other processors. In the language of parallel computing, the computation is said to exhibit poor load balancing. Another example is aerostructural optimization, in which a nonlinear aerodynamics solver may require an order of magnitude more time to run than a linear structural solver [120]. By decomposing the optimization problem, we can balance the processor work loads by allowing discipline analyses with lower computational costs 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 design process occurs not only in parallel but also asynchronously. Overall, this may result in more computational effort, but the intrinsically parallel nature of the architecture allows much of the work to proceed concurrently, reducing the wall-clock time.

B. Classification

We now introduce a new approach to classifying MDO architectures. Some of the previous classifications were based on which constraints could be controlled by the optimizer [77, 121]. Alexandrov and Lewis [121] used the term "closed" to indicate that a set of constraints cannot be satisfied via explicit action of the optimizer, and "open" to indicate that it can. For example, the MDF architecture is closed with respect to both the analysis and consistency constraints, because their satisfaction is determined through the process of converging the MDA. Similarly, IDF is closed with respect to the former but not the latter, since the consistency constraints can be satisfied by the optimizer adjusting the coupling variable copies and design variables. Tosserams et al. [84] expanded on this classification scheme by discussing whether 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 solution via fewer optimization iterations, but this must be weighed against the increased 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 analogs: MDF, IDF, or SAND. This stems from the different approaches to the state and coupling variables in the monolithic architectures. Those distributed architectures that follow MDF and use an MDA (or an approximation of an MDA) to enforce coupling variable consistency at the final solution are classified as distributed MDF architectures. Similarly, those distributed architectures that follow IDF and use coupling variable copies and consistency constraints to enforce consistency at the final solution are classified as distributed IDF architectures.

Our classification 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, our classifica-

tion makes it much easier to see the connections between distributed architectures, even when these architectures are developed in isolation. In many cases, the problem formulation 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, or by applying a specific decomposition scheme. Our classification can also be viewed as a framework in which researchers can develop new distributed architectures, since the starting point for a distributed architecture is *always* a monolithic architecture.

This architecture classification is shown in Fig. 7. The relationships between the architectures are shown by arrows. Some distributed architectures have many variants, such as versions incorporating surrogate models (also known as metamodels or response surfaces) and multiobjective versions; we have included only the core architectures in this diagram. We discuss the different versions of each distributed architecture in the corresponding sections.

No distributed architecture developed to date is an analog of SAND. As discussed in Sec. III, the desire to use independent "black-box" computer software for the discipline analyses necessarily excluded consideration of the SAND 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 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 the coordination uses penalty functions or a problem hierarchy. This grouping follows from the work of de Wit and van Keulen [122].

One area that is not well-explored in MDO is the use of hybrid architectures. A hybrid architecture incorporates elements of two or more architectures in such a way that different discipline 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 [109] and Geethaikrishnan et al. [123]. Such architectures could be especially useful in applications where the coupling characteristics vary widely among the disciplines. However, general rules need to be developed to specify under what conditions the use of certain architectures is advantageous. As we note in Sec. V, much work remains to be done in this area.

The relative performance of the MDO architectures is discussed later, in Sec. V. However, we should make an important remark on benchmarking before describing the various distributed MDO architectures. Because timing results vary based on the hardware and software implementation, the most reliable performance metric when using a gradient-based optimizer is to count the number of times the discipline analyses—and, by extension, the objective and constraint functions—are evaluated. Furthermore, because the discipline analyses often constitute the bulk of the computational work, this metric can serve as a proxy for the total computational time. However, this does not take into account the parallel nature of many computations. Therefore, we should keep in mind that the comparison is less reliable for measuring the wall-clock time in a parallel computing environment.

In the following sections, in chronological order of their initial development, 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 the 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 to explain the advantages and disadvantages of each. Along the way, we will point out variations and applications that can be found in the literature. We also aim to review the state-of-the-art, since the most recent detailed architecture survey in the literature is now more than fifteen years old [66]. More recent surveys, such as that of Agte et al. [56], discuss MDO more generally without discussing the architectures in detail.

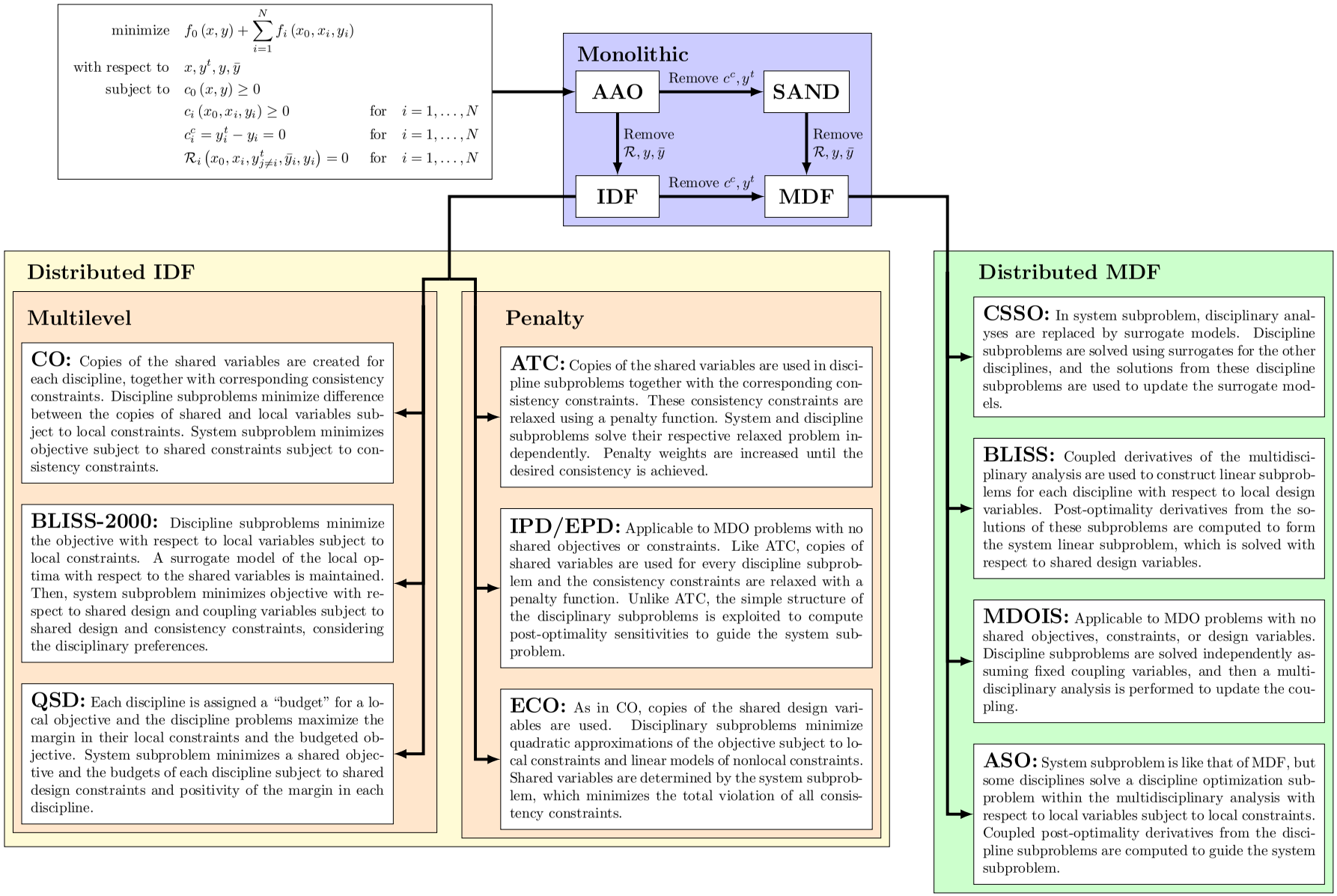

Figure 7. Classification and summary of the MDO architectures.

C. Concurrent Subspace Optimization (CSSO)

CSSO is one of the oldest distributed architectures for large-scale MDO problems. The original formulation [124, 125] 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 an MDA, improving the convergence behavior. At the system level, a coordination problem is solved to recompute the "responsibility," "trade-off," and "switch" coefficients assigned to each discipline to provide information on the design variable preferences for nonlocal constraint satisfaction. Using these coefficients gives each discipline a certain degree of autonomy within the system as a whole. Shankar et al. [126] proposed several improvements to the original architecture, including methods for updating the coefficients, and tested them on two- and three-variable quadratic optimization problems. Unfortunately, they note that the architecture performance is sensitive to parameter selection, and extensive tuning may be required to run CSSO efficiently on larger nonlinear problems.

Several variations of this architecture have been developed to incorporate surrogate models [127, 64, 68] and higher-order information sharing among the disciplines [128]. More recently, the architecture has been adapted to solve multiobjective problems [129, 130, 131]. Parashar and Bloebaum [132] extended a multiobjective CSSO formulation to robust design optimization problems. An application to the design of high-temperature aircraft engine components is presented by Tappeta et al. [133].

The version we consider here, due to Sellar et al. [68], uses surrogate models of each discipline analysis to efficiently model multidisciplinary interactions. Using our unified notation, the CSSO system subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, \tilde{y}\left(x, \tilde{y}\right)\right) \\ \text{with respect to} \quad & x \\ \text{subject to} \quad & c_0\left(x, \tilde{y}\left(x, \tilde{y}\right)\right) \geq 0 \\ & c_i\left(x_0, x_i, \tilde{y}_i\left(x_0, x_i, \tilde{y}_{j\neq i}\right)\right) \geq 0 \quad \text{for} \quad i = 1, \ldots, N \end{aligned} \tag{7}$$

and the discipline i subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y_i\left(x_i, \tilde{y}_{j\neq i}\right), \tilde{y}_{j\neq i}\right) \\ \text{with respect to} \quad & x_0, x_i \\ \text{subject to} \quad & c_0\left(x, \tilde{y}\left(x, \tilde{y}\right)\right) \geq 0 \\ & c_i\left(x_0, x_i, y_i\left(x_0, x_i, \tilde{y}_{j\neq i}\right)\right) \geq 0 \\ & c_j\left(x_0, \tilde{y}_j\left(x_0, \tilde{y}\right)\right) \geq 0 \quad \text{for} \quad j = 1, \ldots, i-1, i+1, \ldots, N \end{aligned} \tag{8}$$

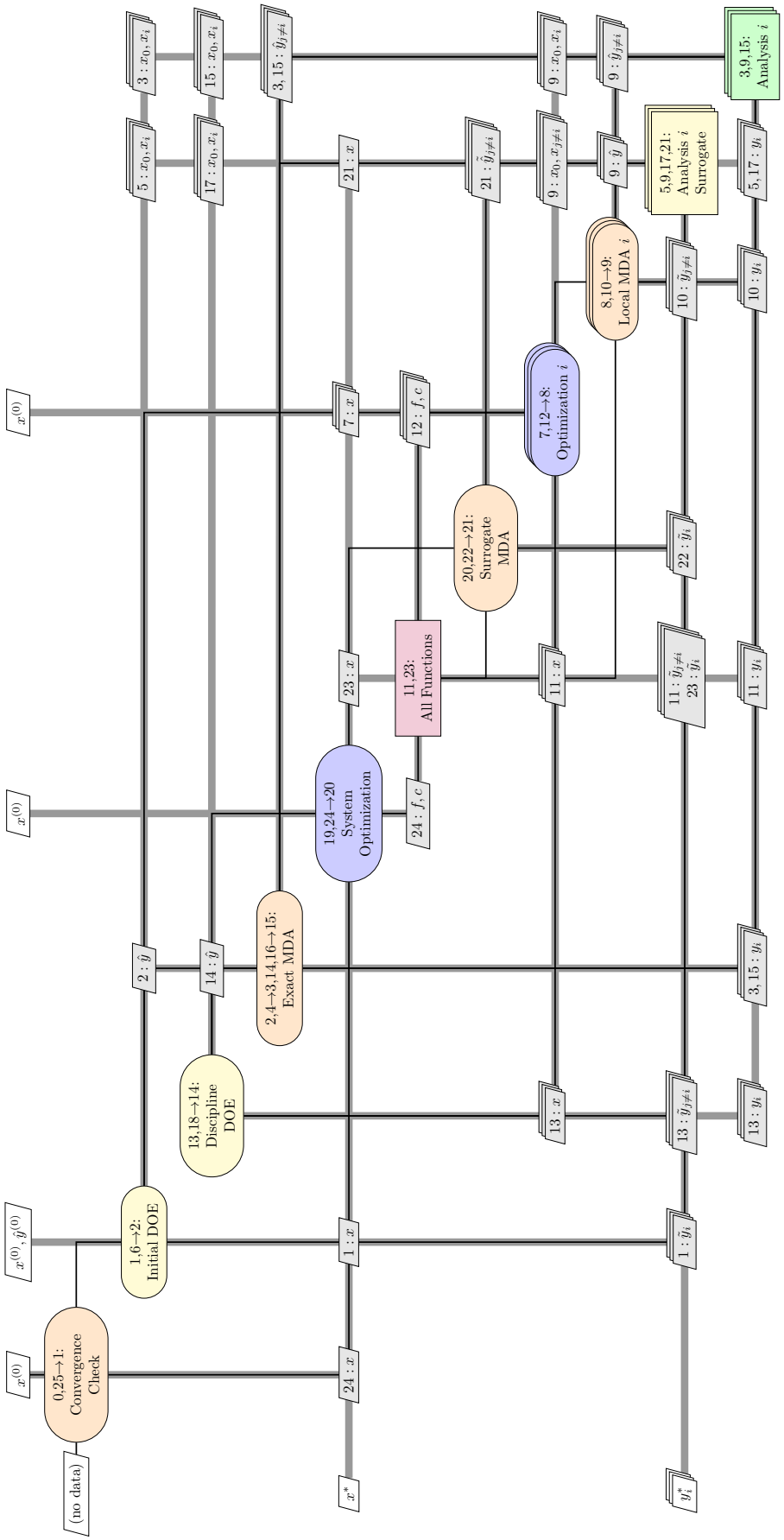

Figure 8. Diagram for the CSSO architecture.

Algorithm 2 Concurrent subspace optimization (CSSO)

Input: Initial design variables $x^{(0)}$

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

0: Initiate main CSSO iteration
repeat
1: Initiate design of experiments (DOE) to generate design points
for each DOE point **do**
2: Initiate MDA that uses exact discipline information
repeat
3: Evaluate discipline analyses
4: Update coupling variables y
until $4 \rightarrow 3$: MDA has converged
5: Update discipline surrogate models with latest design
end for $6 \rightarrow 2$
7: Initiate independent discipline optimizations (in parallel)
for each discipline i **do**
repeat
8: Initiate MDA with exact coupling variables for discipline i and approximate coupling variables for other disciplines
repeat
9: Evaluate discipline i outputs y_i, and surrogate models 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 $12 \rightarrow 8$: Discipline optimization i has converged
end for
13: Initiate DOE that uses subproblem solutions as sample points
for each subproblem solution i **do**
14: Initiate MDA that uses exact discipline information
repeat
15: Evaluate discipline analyses
until $16 \rightarrow 15$ MDA has converged
17: Update discipline surrogate models with newest design
end for $18 \rightarrow 14$
19: Initiate system-level optimization
repeat
20: Initiate MDA that uses only surrogate model information
repeat
21: Evaluate discipline surrogate models
until $22 \rightarrow 21$: MDA has converged
23: Compute objective f_0, and constraint function values c
until $24 \rightarrow 20$: System level problem has converged
until $25 \rightarrow 1$: CSSO has converged

The CSSO architecture is depicted in Fig. 8 and the corresponding steps are listed in Algorithm 2. Note that the architecture uses a surrogate model for every discipline and MDA methods that may, depending on the step in the

algorithm, call the surrogate models directly instead of the discipline analyses. A potential pitfall of this architecture is the need to include all the design variables in the system subproblem. For industrial-scale design problems, this may not always be possible or practical.

There have been some benchmarks that compare CSSO with other MDO architectures. Perez et al. [134], Yi et al. [135], and Tedford and Martins [136] all compare CSSO to other architectures on low-dimensional test problems with gradient-based optimization. Their results show that CSSO required many more analysis calls and function evaluations to converge to an optimal design. The results of de Wit and van Keulen [137] show that CSSO was unable to reach the optimal solution of even a simple minimum-weight two-bar truss problem. Thus, CSSO seems to be largely ineffective when compared with newer MDO architectures.

D. Collaborative Optimization (CO)

In CO, the discipline optimization subproblems are made independent of each other by using copies of the coupling and shared design variables [138, 139]. These copies are then shared with all the disciplines during every iteration of the solution procedure. (In many of the references cited, these variable copies are also known as targets.) The complete independence of the discipline subproblems combined with the simplicity of the data-sharing protocol makes this architecture attractive for problems with a small amount of shared design information.

Braun [138] formulated two versions of the CO architecture: CO_1 and CO_2. CO_2 is more frequently used, so it will be the focus of our discussion. The CO_2 system subproblem is given by

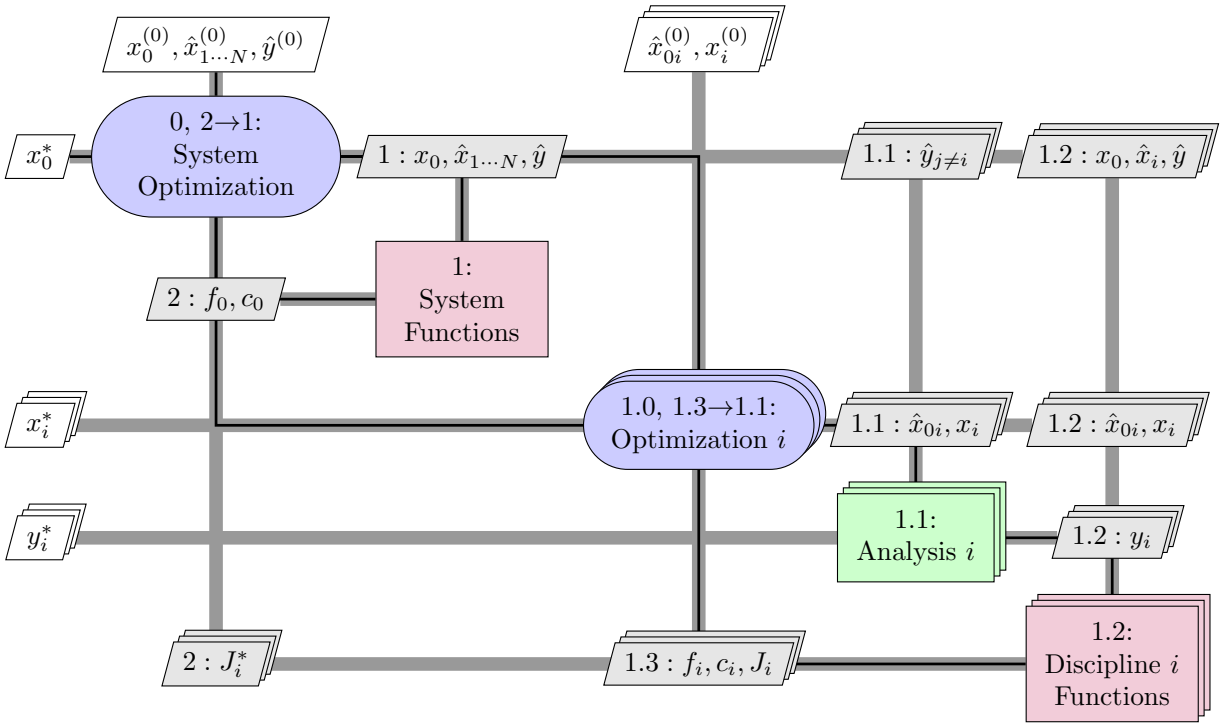

Figure 9. Diagram for the CO architecture.

$$
\begin{aligned}
\text{minimize} \quad & f_0\left(x_0, \hat{x}_1, \ldots, \hat{x}_N, \hat{y}\right) \\
\text{with respect to} \quad & x_0, \hat{x}_1, \ldots, \hat{x}_N, \hat{y} \\
\text{subject to} \quad & c_0\left(x_0, \hat{x}_1, \ldots, \hat{x}_N, \hat{y}\right) \geq 0 \\
& J_i^* = ||\hat{x}_{0i} - x_0||_2^2 + ||\hat{x}_i - x_i||_2^2 + \\
& \qquad ||\hat{y}_i - y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j \neq i}\right)||_2^2 = 0 \quad \text{for} \quad i = 1, \ldots, N
\end{aligned}
\tag{9}
$$

where the $\hat{x}_{0i}$ are copies of the global design variables passed to—and manipulated by—discipline i, and the $\hat{x}_i$ are copies of the local design variables passed to the system subproblem. These copies are independent variables whose values are chosen by a different subproblem. Equality constraints are used to ensure that both copies agree on a single value at an optimal design. Copies of the local design variables are made only if those variables directly influence the objective. In CO_1, the quadratic equality constraints are replaced with linear equality constraints for each shared variable and its copy. In either case, if derivatives are required to solve the system subproblem, they must be computed with respect to the *optimized* function J_i^*. Although the architecture has yet to converge on an optimized system design, this step is referred to as a post-optimality analysis because the subsystems have been optimized with respect to their local information.

The discipline i subproblem in both CO_1 and CO_2 is

$$
\begin{aligned}
\text{minimize} \quad & J_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j \neq i}\right)\right) \\
\text{with respect to} \quad & \hat{x}_{0i}, x_i \\
\text{subject to} \quad & c_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j \neq i}\right)\right) \geq 0.
\end{aligned}
\tag{10}
$$

Thus, the system-level problem is responsible for minimizing the design objective, while the discipline-level problems minimize system inconsistency. Braun [138] showed that the CO problem statement is mathematically equivalent to the IDF problem statement (3) and therefore equivalent to the original MDO problem (1) as well. In particular, if the CO architecture converges to a point that locally minimizes f_0 while satisfying the design constraints c_0 and c_i and consistency constraints $J_i = 0$, the resulting point must also be a local minimum of the IDF problem. This can be inferred from the special structure of problems (9) and (10). CO is depicted by the XDSM in Fig. 9. The corresponding procedure is detailed in Algorithm 3.

Algorithm 3 Collaborative optimization (CO)

Input: Initial design variables $x^{(0)}$

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 discipline optimization

repeat

1.1: Evaluate discipline analysis

1.2: Compute discipline subproblem objective and constraints

1.3: Compute new discipline subproblem design point and J_i

until $1.3 \rightarrow 1.1$: Optimization i has converged

end for

2: Compute new system subproblem design point

until $2 \rightarrow 1$: System optimization has converged

In spite of the organizational advantage of fully separate discipline subproblems, CO has major weaknesses in the mathematical formulation that lead to poor performance in practice [61, 140]. In particular, the system problem in CO_1 has more equality constraints than variables, so the system subproblem is often 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 [61]. 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 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 [48], or a gradient-based optimizer that handles the troublesome constraints by avoiding the use of Lagrange multipliers in the termination condition [141]. While such approaches do avoid the obvious problems with CO, they introduce other issues. Gradient-free optimizers that do not employ some kind of surrogate modeling tend to require many more function evaluations than do gradient-based optimizers. These additional function evaluations and discipline analyses can become a bottleneck. Gradient-based optimizers that avoid Lagrange multipliers, such as feasible direction methods, often fail in nonconvex feasible regions. As pointed out by DeMiguel [140], the CO system subproblem is set-constrained, i.e., nonconvex, because of the need to satisfy optimality in the discipline subproblems.

Several researchers successfully use CO together with robust gradient-based optimization methods. DeMiguel and Murray [140] relax the troublesome constraints using an L_1 exact penalty function with a fixed penalty parameter value, and they 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, since algorithms using the penalty function formulation are known to converge to an optimal solution under mild assumptions [142, 42]. However, the results of Brown and Olds [143] show strange behavior in a practical design problem. In particular, when the penalty parameter was above a threshold value, the algorithm could not improve on the initial design point. Below a lower threshold value, the architecture showed poor convergence. Finally, the authors could not find a penalty parameter that produced a final design close to those computed by other architectures. In light of these findings, they did not test MCO further.

Another idea, proposed by Sobieski and Kroo [74], uses surrogate models to approximate the post-optimality behavior of the discipline subproblems in the system subproblem. This both eliminates the direct calculation of post-optimality derivatives 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 CO adjustment involves relaxing the system subproblem equality constraints to inequalities with a relaxation tolerance; this was originally proposed by Braun et al. [139]. This approach was also successful in other test problems [144, 145], where the 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. [146] build on this approach by adaptively choosing the tolerance during the solution procedure so that the system subproblem remains feasible at each iteration. This approach appears to work for the test problems in [61] 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, mostly in the design of aerospace systems. Examples include the design of launch vehicles [39], rocket engines [40], satellite constellations [147], flight trajectories [72, 148], and flight control systems [149], as well as the preliminary design of complete aircraft [17, 18] and aircraft family design [150]. Beyond aerospace engineering, CO has been applied to problems involving automobile engines [28], bridge design [22], and railway cars [26], and even the design of a scanning optical microscope [27].

Adaptations of the CO architecture have also been developed for multiobjective, robust, and multifidelity MDO problems. Multiobjective formulations were first described by Tappeta and Renaud [43]. McAllister et al. [151] present a multiobjective approach using linear physical programming. Available robust design formulations incorporate the decision-based models of Gu et al. [152] and McAllister and Simpson [28], the implicit uncertainty propagation method of Gu et al. [153], and the fuzzy computing models of Huang et al. [154]. Zadeh and Toropov [118] integrated multiple model fidelities into CO for an aircraft design problem.

The most recent version of CO—enhanced collaborative optimization (ECO)—was developed by Roth and Kroo [155, 55]. Figure 10 shows the XDSM corresponding to this architecture. ECO, while still derived from the same basic problem as the original CO architecture, is radically different and therefore deserves attention. In a sense, the roles of the system and discipline optimization have been reversed in ECO: the system subproblem minimizes system infeasibility, while the discipline subproblems minimize the system objective. The system subproblem is

$$\begin{aligned} \text{minimize} \quad & J_0 = \sum_{i=1}^{N} ||\hat{x}_{0i} - x_0||_2^2 + ||\hat{y}_i - y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right)||_2^2 \\ \text{with respect to} \quad & x_0, \hat{y}. \end{aligned} \tag{11}$$

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

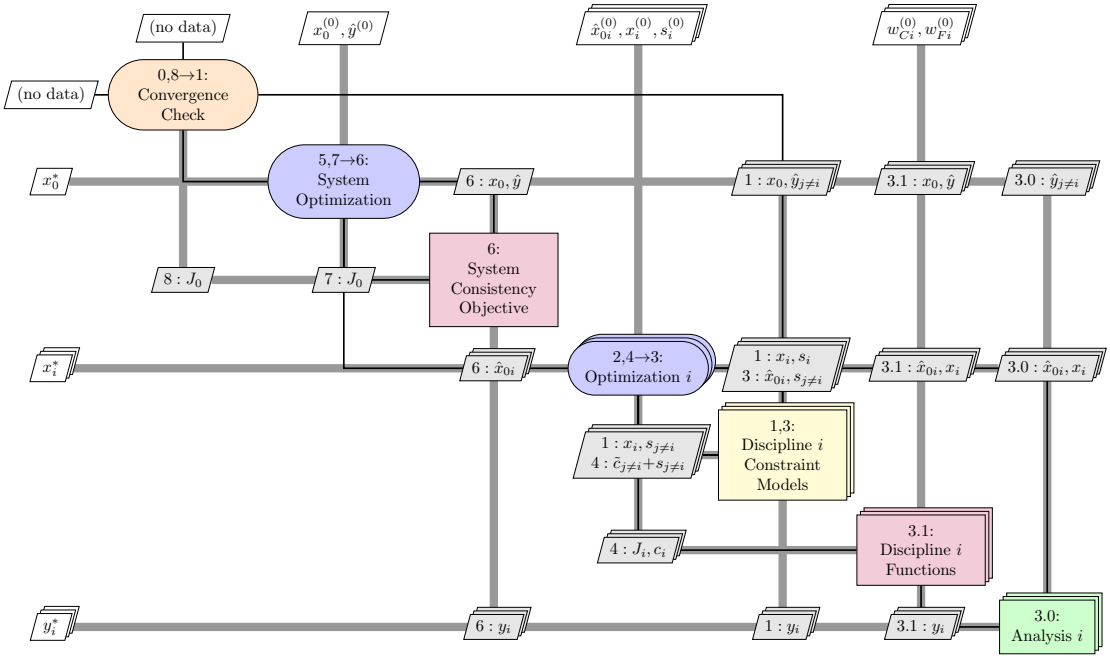

Figure 10. XDSM for the ECO architecture.

The i^{th} discipline subproblem is

$$\begin{aligned}
\text{minimize} \quad & J_i = \tilde{f}_0\left(\hat{x}_{0i}, y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i}\right)\right) + \\
& w_{Ci}\left(||\hat{x}_{0i} - x_0||_2^2 + ||\hat{y}_i - y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i}\right)||_2^2\right) + \\
& w_{Fi}\sum_{j=1, j\neq i}^{N}\sum_{k=1}^{n_s} s_{jk} \\
\text{with respect to} \quad & \hat{x}_{0i}, x_i, s_{j\neq i} \\
\text{subject to} \quad & c_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i}\right)\right) \geq 0 \\
& \tilde{c}_j\left(\hat{x}_{0i}\right) + s_j \geq 0 \qquad\qquad j = 1, \ldots, i-1, i+1, \ldots, N \\
& s_j \geq 0 \qquad\qquad j = 1, \ldots, i-1, i+1, \ldots, N,
\end{aligned} \tag{12}$$

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 [55].

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, represented by $\tilde{f}_0$, in each discipline subproblem. This is meant to increase each discipline's "awareness" of its 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 Fig. 10. The constraint models for each discipline are constructed by first solving the optimization problem that minimizes the constraint violation with respect to the local elastic and design variables, i.e.,

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{n_s} s_{ik} \\
\text{with respect to} \quad & x_i, s_i \\
\text{subject to} \quad & c_i\left(x_0, x_i, y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right)\right) + s_i \geq 0 \\
& s_i \geq 0,
\end{aligned} \tag{13}$$

where the shared design variables and the coupling variable copies are treated as fixed parameters. Post-optimality derivatives are then computed to determine the change in the optimized local design variables with respect to the change in the 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 (13) are then used as part of the initial data for Problem (12). The full algorithm for ECO is listed in Algorithm 4.

Algorithm 4 Enhanced collaborative optimization (ECO)

Input: Initial design variables $x^{(0)}$

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 discipline optimization
    repeat
      3: Evaluate nonlocal constraint models with local copies of shared variables
      3.0: Evaluate discipline analysis
      3.1: Compute discipline subproblem objective and constraints
      4: Compute new discipline subproblem design point and J_i
    until $4 \rightarrow 3$: Discipline subproblem has converged
  end for
  5: Initiate system optimization
  repeat
    6: Compute J_0
    7: Compute updated values of x_0 and $\hat{y}$.
  until $7 \rightarrow 6$: System optimization has converged
until $8 \rightarrow 1$: J_0 is below specified tolerance

Based on Roth's results [155, 55], 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 describe in Sec. F.

While ECO seems to be effective, CO tends to be an inefficient architecture for MDO problems. Without the modifications discussed in this section, the architecture requires a disproportionately large number of function and discipline evaluations [156, 157, 137, 135], assuming it converges at all. When the system-level equality constraints are relaxed, the results from CO are more competitive with those from other distributed architectures [134, 145, 136] but still compare poorly with those from monolithic architectures.

E. Bilevel Integrated System Synthesis (BLISS)

The BLISS architecture [69], like CSSO, is a method for decomposing the MDF problem along discipline lines. Unlike CSSO, however, BLISS assigns local design variables to discipline 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 [158]. These approximations are constructed at each iteration using coupled sensitivity

information. The system-level subproblem is

$$\begin{aligned}
\text{minimize} \quad & (f_0^*)_0 + \left(\frac{\mathrm{d}f_0^*}{\mathrm{d}x_0}\right)\Delta x_0 \\
\text{with respect to} \quad & \Delta x_0 \\
\text{subject to} \quad & (c_0^*)_0 + \left(\frac{\mathrm{d}c_0^*}{\mathrm{d}x_0}\right)\Delta x_0 \geq 0 \\
& (c_i^*)_0 + \left(\frac{\mathrm{d}c_i^*}{\mathrm{d}x_0}\right)\Delta x_0 \geq 0 \quad \text{for} \quad i = 1, \ldots, N \\
& \Delta x_{0L} \leq \Delta x_0 \leq \Delta x_{0U}.
\end{aligned} \tag{14}$$

The discipline i subproblem is given by

$$\begin{aligned}
\text{minimize} \quad & (f_0)_0 + \left(\frac{\mathrm{d}f_0}{\mathrm{d}x_i}\right)\Delta x_i \\
\text{with respect to} \quad & \Delta x_i \\
\text{subject to} \quad & (c_0)_0 + \left(\frac{\mathrm{d}c_0}{\mathrm{d}x_i}\right)\Delta x_i \geq 0 \\
& (c_i)_0 + \left(\frac{\mathrm{d}c_i}{\mathrm{d}x_i}\right)\Delta x_i \geq 0 \\
& \Delta x_{iL} \leq \Delta x_i \leq \Delta x_{iU}.
\end{aligned} \tag{15}$$

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

To prevent violation of the discipline constraints by changes in the shared design variables, post-optimality derivative information (the change in the optimized discipline constraints with respect to a change in the system design variables) is required to solve the system subproblem. For this step, Sobieski [69] presents two methods: one based on a generalized version of the global sensitivity equations [107], 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 surrogate models to compute post-optimality derivatives [67, 159].

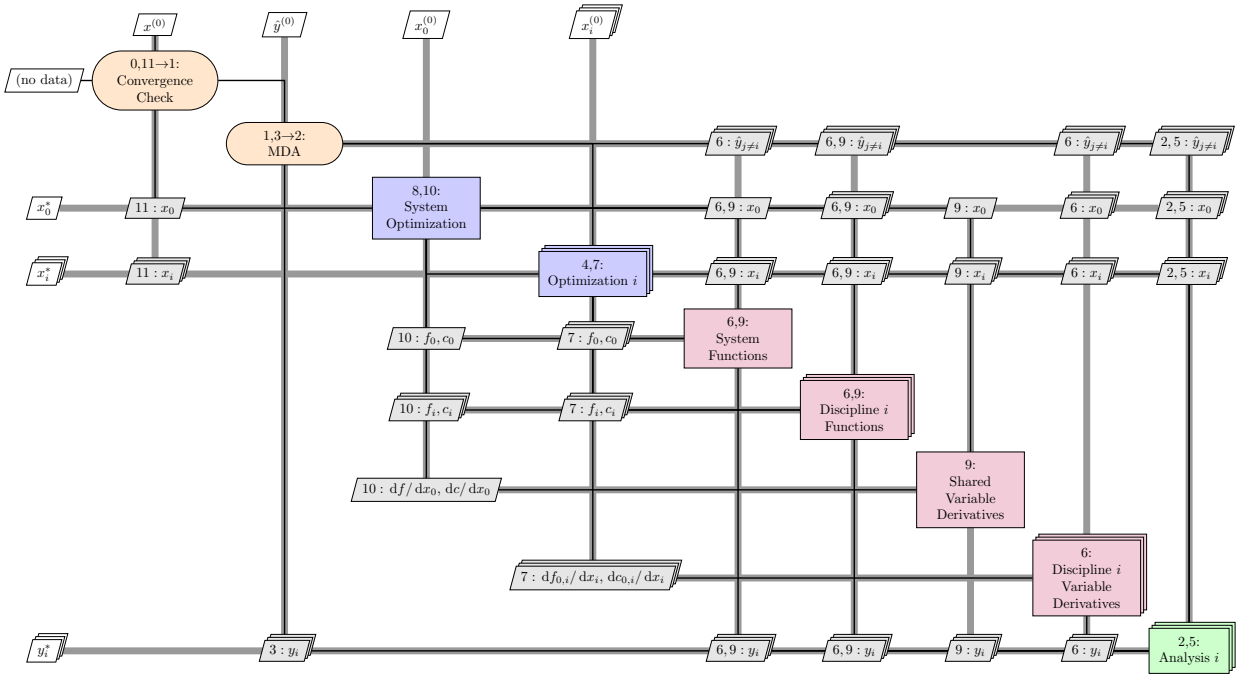

Figure 11. Diagram for the BLISS architecture.

Algorithm 5 Bilevel system integrated system synthesis (BLISS)

Input: Initial design variables $x^{(0)}$

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 optimal solution for discipline 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 analysis

10: Compute optimal solution to system subproblem

until $11 \rightarrow 1$: System optimization has converged

Figure 11 shows the XDSM for BLISS, and the procedure is listed in Algorithm 5. Because of the linear nature

of the optimization problems under consideration, repeated evaluation 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, e.g., through a trust-region framework. Detailed knowledge of the design space can also help, but this increases the implementation overhead.

There are two adaptations of the original BLISS architecture. The first is Ahn and Kwon's proBLISS [160], 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 [161], an architecture that integrates a reduced-order modeling technique called proper orthogonal decomposition [162] to reduce the cost of the MDA and gradient computation steps. Their results show a significant improvement in the performance of BLISS, to the point where it is almost competitive with MDF.

A radically different formulation called BLISS-2000 was developed by Sobieski et al. [53]. Because BLISS-2000 does not require an MDA to restore the feasibility of the design, we have separated it from the other BLISS variants in the classification shown in Fig. 7. In fact, like other IDF-derived architectures, BLISS-2000 uses coupling variable copies to enforce consistency at the optimum. Information exchange between the system and discipline subproblems occurs through surrogate models of the discipline optima. The BLISS-2000 system subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, \tilde{y}\left(x, \hat{y}\right)\right) \\ \text{with respect to} \quad & x_0, \hat{y}, w \\ \text{subject to} \quad & c_0\left(x, \tilde{y}\left(x, \hat{y}, w\right)\right) \geq 0 \\ & \hat{y}_i - \tilde{y}_i\left(x_0, x_i, \hat{y}_{j\neq i}, w_i\right) = 0 \quad \text{for} \quad i = 1, \ldots, N. \end{aligned} \tag{16}$$

The BLISS-2000 discipline i subproblem is

$$\begin{aligned} \text{minimize} \quad & w_i^T y_i \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & c_i\left(x_0, x_i, y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{17}$$

A unique aspect of this architecture is the use of a vector of weighting coefficients, w_i, attached to the discipline states. These weighting coefficients give the user a measure of control over the state variable preferences. Generally speaking, the coefficients should be chosen based on the structure of the global objective to allow the discipline subproblems to find an optimum more quickly. The impact of these coefficients on convergence has yet to be determined. The XDSM for the architecture is shown in Fig. 12, and the procedure is listed in Algorithm 6.

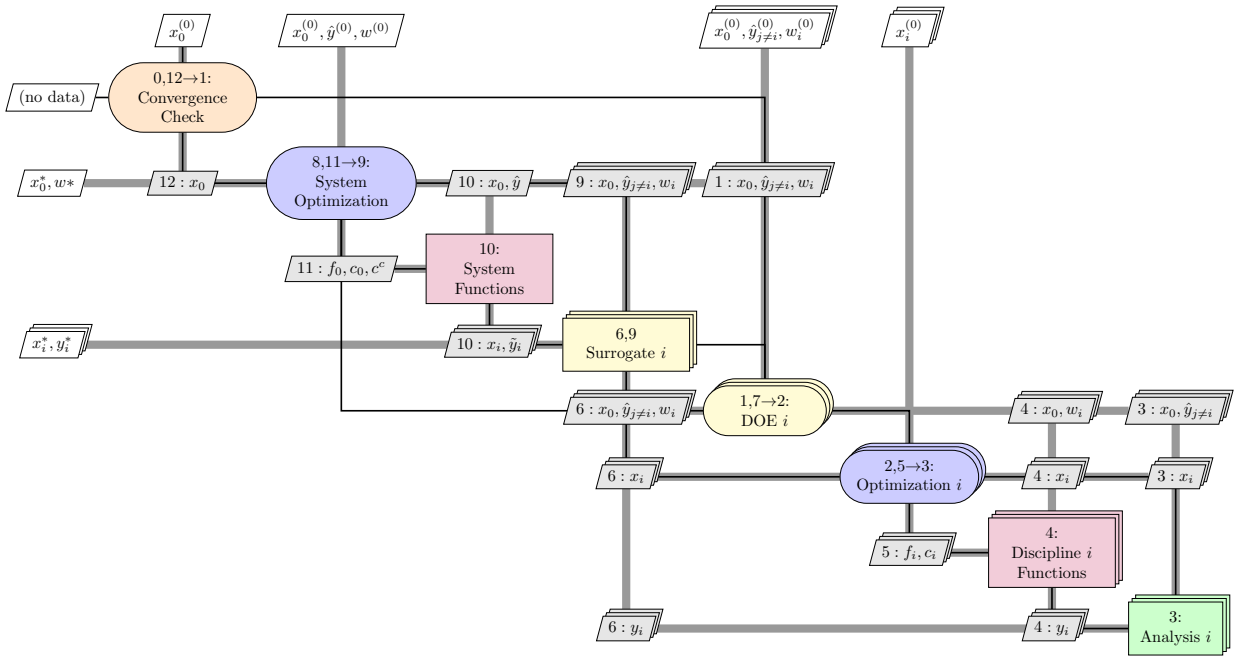

Figure 12. Diagram for the BLISS-2000 architecture.

Algorithm 6 BLISS-2000

Input: Initial design variables $x^{(0)}$

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

0: Initiate main BLISS iteration

repeat

  for each discipline i **do**

    1: Initiate DOE

    for each DOE point **do**

      2: Initiate discipline optimization

      repeat

        3: Evaluate discipline analysis

        4: Compute discipline objective f_i and constraint functions c_i

        5: Update local design variables x_i

      until $5 \rightarrow 3$: Discipline optimization subproblem has converged

      6: Update surrogate model of optimized discipline subproblem with new solution

    end for $7 \rightarrow 1$

  end for

  8: Initiate system optimization

  repeat

    9: Evaluate surrogate models with current values of system variables

    10: Compute system objective and constraint function values

    11: Compute new system design point

  until $11 \rightarrow 9$ System subproblem has converged

until $12 \rightarrow 1$: System optimization has converged

BLISS-2000 has several advantages over the original BLISS architecture. First, the solution procedure is much easier to understand, and this simplicity greatly reduces the obstacles to implementation. Second, the decomposed problem formulation is equivalent to Problem (1) provided the weighting coefficients accurately capture the influence of the coupling variables on the objective [53]. Third, by using surrogate models 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 [163]. 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 original formulation. Many of the results available [134, 137, 135] 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 derivative 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 [143]. In this case, BLISS-2000 tended to outperform other distributed architectures, and it 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.

F. 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 [164, 57]. If the targets are unattainable, the ATC architecture returns a design point that minimizes the unattainability. 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 the 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. [165]. This formulation conforms to our definition of an MDO problem by explicitly including system-wide objective and constraint functions. Like all the other ATC problem formulations, a solution generated by this architecture will solve Problem (1) provided the penalty terms in the optimization problems all approach zero. The ATC system subproblem is given by

$$\begin{aligned} \text{minimize} \quad & f_0\left(x,\hat{y}\right)+\sum_{i=1}^{N}\Phi_i\left(\hat{x}_{0i}-x_0,\hat{y}_i-y_i\left(x_0,x_i,\hat{y}\right)\right)+ \\ & \Phi_0\left(c_0\left(x,\hat{y}\right)\right) \\ \text{with respect to} \quad & x_0,\hat{y}, \end{aligned} \tag{18}$$

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 variable copies $\hat{x}_{0i}$ have the same function as in the CO architecture, with the penalty function used to ensure all the copies converge on the same value in an optimal design. The i^{th} discipline subproblem is

$$\begin{aligned} \text{minimize} \quad & f_0\left(\hat{x}_{0i},x_i,y_i\left(\hat{x}_{0i},x_i,\hat{y}_{j\neq i}\right),\hat{y}_{j\neq i}\right)+f_i\left(\hat{x}_{0i},x_i,y_i\left(\hat{x}_{0i},x_i,\hat{y}_{j\neq i}\right)\right)+ \\ & \Phi_i\left(\hat{y}_i-y_i\left(\hat{x}_{0i},x_i,\hat{y}_{j\neq i}\right),\hat{x}_{0i}-x_0\right)+ \\ & \Phi_0\left(c_0\left(\hat{x}_{0i},x_i,y_i\left(\hat{x}_{0i},x_i,\hat{y}_{j\neq i}\right),\hat{y}_{j\neq i}\right)\right) \\ \text{with respect to} \quad & \hat{x}_{0i},x_i \\ \text{subject to} \quad & c_i\left(\hat{x}_{0i},x_i,y_i\left(\hat{x}_{0i},x_i,\hat{y}_{j\neq i}\right)\right)\geq 0. \end{aligned} \tag{19}$$

Figure 13 shows the ATC architecture XDSM, where w denotes the penalty function weights used in the determination

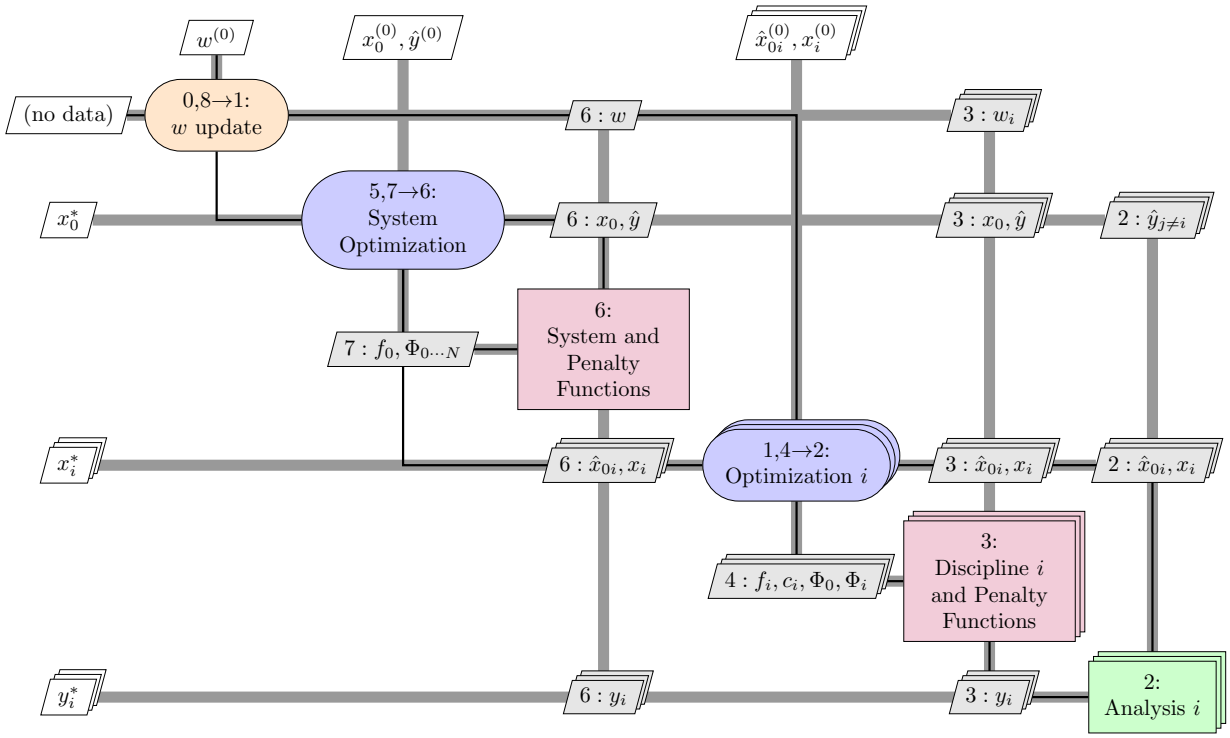

Figure 13. Diagram for the ATC architecture.

of Φ_0 and Φ_i. The details of ATC are described in Algorithm 7.

Algorithm 7 Analytical target cascading (ATC)

Input: Initial design variables $x^{(0)}$

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 discipline analysis
      3: Compute discipline objective and constraint functions and penalty function values
      4: Update discipline design variables
    until 4 → 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 variable copies
  until 7 → 6: System optimization has converged
  8: Update penalty weights
until 8 → 1: System optimization has converged
```

ATC can be applied to a multilevel hierarchy of systems as well as to a discipline-based nonhierarchical 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. Post-optimality derivatives are not needed in any of the subproblems, since nonlocal data are always treated as fixed values in the current subproblem. Furthermore, as with penalty methods for general optimization problems (see, e.g., Nocedal and Wright [42, 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 because of the nonseparable nature of the quadratic penalty function.

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 consistency in the discipline models and convergence of the algorithm. Michalek and Papalambros [59] present an effective weight-update method that is especially useful when unattainable targets have been set in a traditional ATC process. Michelena et al. [58] present several coordination algorithms using ATC with quadratic penalty functions and demonstrate the convergence to an optimum for all of them. However, the analysis of Michelena et al. is based on a top-level target satisfaction objective function that is convex. The behavior of these versions of ATC in the presence of nonconvex objectives is unknown.

Several other penalty function choices and associated coordination approaches have been devised for ATC. Kim et al. [166] outline a version of ATC that uses Lagrangian relaxation and a subgradient method to update the multiplier estimates. Tosserams et al. [167] use augmented Lagrangian relaxation with Bertsekas' method of multipliers [168] and the alternating-direction method of multipliers [117] to update the penalty weights. They also group this variant of ATC into a larger class of coordination algorithms known as augmented Lagrangian coordination [165]. Li et al. [169] apply the diagonal quadratic approximation approach of Ruszczynski [170] to the augmented Lagrangian to eliminate subproblem coupling through the quadratic terms and further parallelize the architecture. Finally, Han and Papalambros [171] propose a version of ATC based on sequential linear programming [172, 173], where inconsistency is penalized using infinity norms. They later presented a convergence proof of this approach in a short note [174].

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. It has been most frequently applied to design problems in the field for which it was developed, the automotive industry [175, 176, 177, 178, 29, 179, 180, 181]. However, it has also proved useful in aircraft design [182, 150, 54] and building design [23]. Applications beyond engineering design include manufacturing decisions [183], supply chain management [184], and marketing decisions in product design [185]. Huang et al. [186] have developed an ATC-specific web portal to solve optimization problems via the ATC architecture. Etman et al. [187] discuss the automatic implementation of coordination procedures, using ATC as an example architecture. Formulations exist that can handle integer variables [47] and probabilistic design problems [188, 189]. The method has also been adapted for problems with block-separable linking constraints [190]. In this class of problems, c_0 consists of constraints that 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; only one result is available. In de Wit and Van Keulen's architecture comparison [137], ATC is competitive with all the other benchmarked distributed architectures, including standard versions of CSSO, CO, and BLISS. However, with a gradient-based optimizer, ATC and the other distributed architectures are not competitive with a monolithic architecture in terms of the number of function and discipline evaluations. Different versions of ATC have been benchmarked against each other. Tosserams et al. [167] compared the augmented Lagrangian penalty approach with the quadratic penalty approach and found much improved results with the alternating-direction method of multipliers. Surprisingly, de Wit and van Keulen [137] found that the augmented Lagrangian version performed worse than the quadratic penalty method for their test problem. Han and Papalambros [171] 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 this needs to be addressed.

G. 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) [191, 71] may be employed. Both formulations rely on solving the discipline subproblem

$$\begin{aligned} \text{minimize} \quad & f_i\left(\hat{x}_{0i}, x_i, y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i}\right)\right) + \Phi_i\left(\hat{x}_{0i} - x_0, \hat{y}_i - y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i}\right)\right) \\ \text{with respect to} \quad & \hat{x}_{0i}, x_i \\ \text{subject to} \quad & c_i\left(\hat{x}_{0i}, x_i, y_i\left(x_{0i}, x_i, \hat{y}_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{20}$$

Here, Φ_i denotes the penalty function associated with the inconsistency between the i^{th} discipline's 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 original variable. As in the ATC architecture, the penalty function is used to ensure that all variable copies converge to a single value in an optimal design.

At the system level, the subproblem is an unconstrained minimization with respect to the shared variables and coupling variable copies. The objective function is the sum of the optimized discipline penalty terms, denoted as Φ_i^*:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} \Phi_i^*\left(x_0, \hat{y}\right) = \sum_{i=1}^{N} \Phi_i\left(\hat{x}_{0i} - x_0, \hat{y}_i - y_i\left(\hat{x}_{0i}, x_i, \hat{y}_{j\neq i}\right)\right) \\ \text{with respect to} \quad & x_0, \hat{y}. \end{aligned} \tag{21}$$

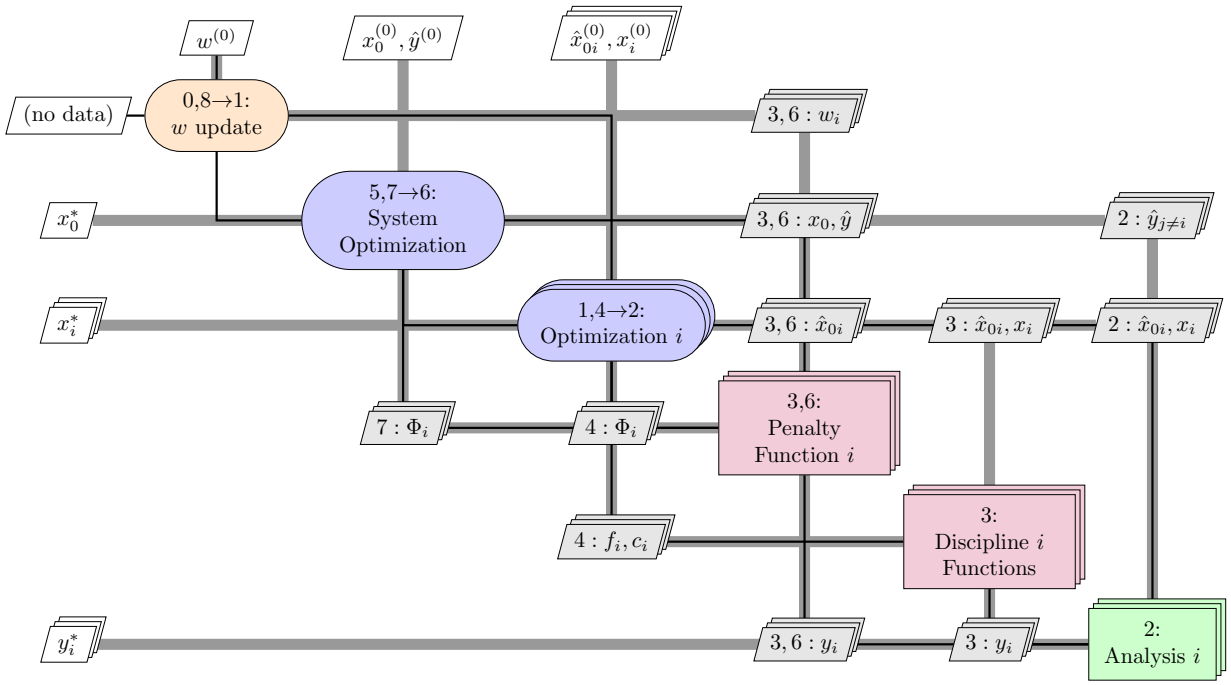

Figure 14. Diagram for the penalty decomposition architectures EPD and IPD.

The penalty weights are updated upon solution of the system problem. Figure 14 shows the XDSM for this architecture, where w represents the penalty weights. The sequence of operations in this architecture is detailed in Algorithm 8.

Algorithm 8 Exact and inexact penalty decomposition (EPD/IPD)

Input: Initial design variables $x^{(0)}$

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 variable copies
  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 to a KKT point under the linear independence constraint qualification and with mild assumptions on the update strategy for the penalty weights [71]. In particular, similarly to other quadratic penalty methods, the penalty weight in IPD must monotonically increase until the inconsistency is sufficiently small [42]. For EPD, the penalty weight must be larger than the largest Lagrange multiplier, following the established theory for the L_1 penalty function [42], while the barrier parameter must monotonically decrease as in an interior point method [106]. 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 (1).

Only once in the literature has either architecture been tested against others. The results of Tosserams et al. [192] suggest that the performance depends on the penalty function employed. A comparison of IPD with a quadratic penalty function and IPD with an augmented Lagrangian penalty function showed that the latter significantly outperformed the former on several test problems in terms of both time and number of function evaluations.

H. MDO of Independent Subspaces (MDOIS)

If the problem contains no system-wide constraints or objectives, i.e., if f_0 and c_0 do not 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 [70] applies. In this case, the discipline subproblems are fully separable (aside from the coupled state variables) and given by

$$\begin{aligned} \text{minimize} \quad & f_i\left(x_i, y_i\left(x_i, \hat{y}_{j\neq i}\right)\right) \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & c_i\left(x_i, y_i\left(x_i, \hat{y}_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{22}$$

The coupling variable copies are just parameters representing the system state information. Upon solution of the

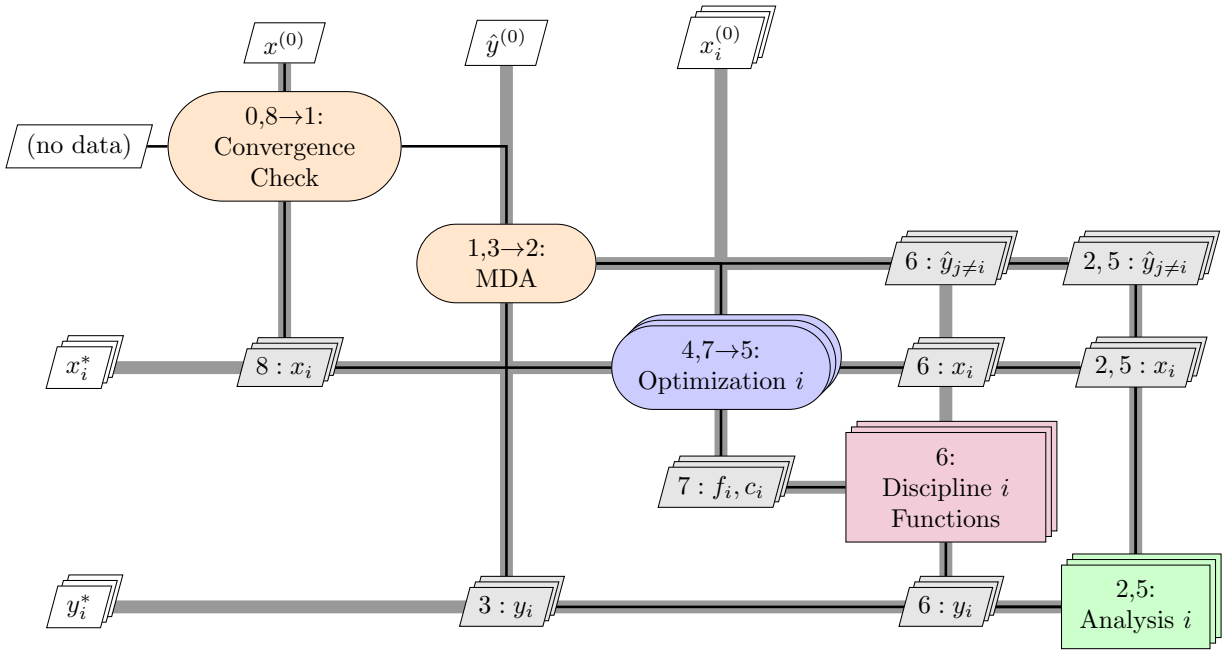

Figure 15. Diagram for the MDOIS architecture.

discipline subproblems, which can access the output of individual discipline analysis codes, a full MDA is completed to update all parameter values. Thus, rather than the system subproblem used by other architectures, the MDA is used to guide the discipline subproblems to a design solution. Shin and Park [70] show that under the given problem assumptions an optimal design of the original (MDF) problem is found using this architecture. Figure 15 depicts this process in an XDSM. MDOIS is detailed in Algorithm 9.

Algorithm 9 Multidisciplinary design optimization of independent subspaces (MDOIS)

Input: Initial design variables $x^{(0)}$

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 $3 \rightarrow 2$: MDA has converged
  for each discipline i **do**
    4: Initiate discipline optimization
    repeat
      5: Evaluate discipline analysis
      6: Compute discipline objectives and constraints
      7: Compute new discipline design point
    until $7 \rightarrow 5$: Discipline optimization has converged
  end for
until $8 \rightarrow 1$ Main iteration has converged

Benchmarking results are available that compare MDOIS to some of the older architectures. These results are given by Yi et al. [135]. In many cases, MDOIS requires fewer analysis calls than MDF while still being able to reach the optimal solution. However, MDOIS 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. [193]. However, the results have not been compared to those obtained using other architectures.

I. Quasiseparable Decomposition (QSD)

Haftka and Watson [65] 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 is equivalent to the complicating-variable problems discussed in Sec. A. We have not come across any practical design problems that do not satisfy this property. However, if necessary, we can easily transform the general MDO problem (1) into a quasiseparable problem. This is accomplished by copying the relevant local variables and forcing the global objective to depend on the copies. (The process is analogous to adapting the general MDO problem to the original CO architecture.) By using a local equality constraint to ensure that the copied variables converge to the same value, we ensure that a solution to the quasiseparable

problem is also a solution to the original (IDF) problem. The system subproblem of QSD is given by

$$\begin{aligned}
\text{minimize} \quad & f_0\left(x_0, \hat{y}\right) + \sum_{i=1}^{N} b_i \\
\text{with respect to} \quad & x_0, \hat{y}, b \\
\text{subject to} \quad & c_0\left(x_0, \hat{y}\right) \geq 0 \\
& s_i^*\left(x_0, x_i, y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right), b_i\right) \geq 0 \quad \text{for} \quad i = 1, \ldots, N
\end{aligned} \tag{23}$$

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

$$\begin{aligned}
\text{minimize} \quad & -s_i \\
\text{with respect to} \quad & x_i, s_i \\
\text{subject to} \quad & \bar{c}_{i_k} = c_{i_k}\left(x_0, x_i, y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right)\right) - s_i \geq 0 \quad \text{for} \quad k = 1, \ldots, m_i \\
& \bar{f}_i = b_i - f_i\left(x_0, x_i, y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right)\right) - s_i \geq 0 \\
& c^c = \hat{y}_i - y_i\left(x_0, x_i, \hat{y}_{j\neq i}\right) = 0
\end{aligned} \tag{24}$$

where k is an element of the constraint vector c_i. Because of the use of coupling variable copies, we classify this architecture as distributed IDF. Similarly to CO, this is a bilevel architecture where the solutions of the discipline subproblems are constraints in the system subproblem. Therefore, post-optimality derivatives or surrogate model approximations of the optimized discipline subproblems are required to solve the system subproblem. The standard architecture is shown as an XDSM in Fig. 16. The sequence of operations in QSD is given by Algorithm 10. Haftka and Watson have extended the theory behind QSD to problems with a combination of discrete and continuous variables [46]. Liu et al. [194] successfully applied QSD with surrogate models to a structural optimization problem. However, they made no comparisons to other architectures, not even QSD without surrogates. A version of QSD without surrogate models was benchmarked by de Wit and van Keulen [137]. Unfortunately, this architecture was the worst of all the architectures tested in terms of discipline evaluations. A version using surrogate models should yield improved performance, because of the smoothness introduced by the model, but this version has not to our knowledge been benchmarked.

Algorithm 10 Quasiseparable decomposition (QSD)

Input: Initial design variables $x^{(0)}$

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 new system design point

until $2 \rightarrow 1$: System problem has converged

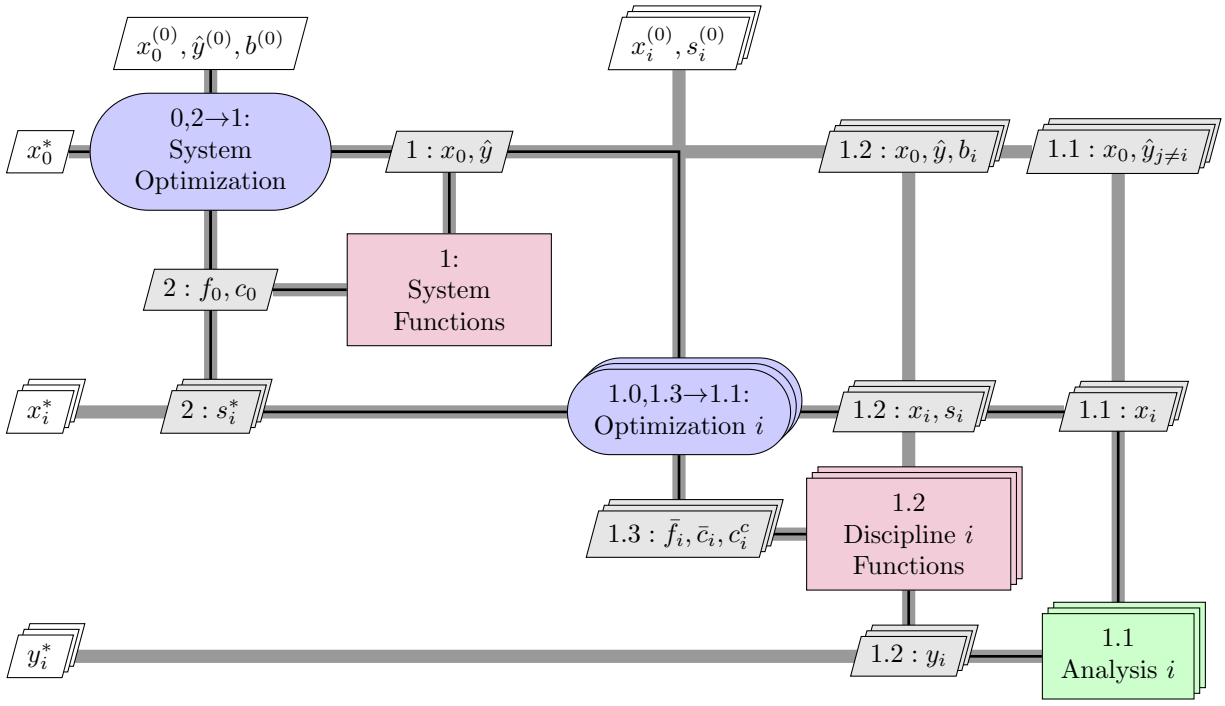

Figure 16. Diagram for the QSD architecture.

J. Asymmetric Subspace Optimization (ASO)

The ASO architecture [120] is a new distributed-MDF architecture. It was motivated by high-fidelity aerostructural optimization, where the aerodynamic analysis typically requires an order of magnitude more time to complete than the structural analysis [195]. 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 the discipline analysis times. Figure 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 11.

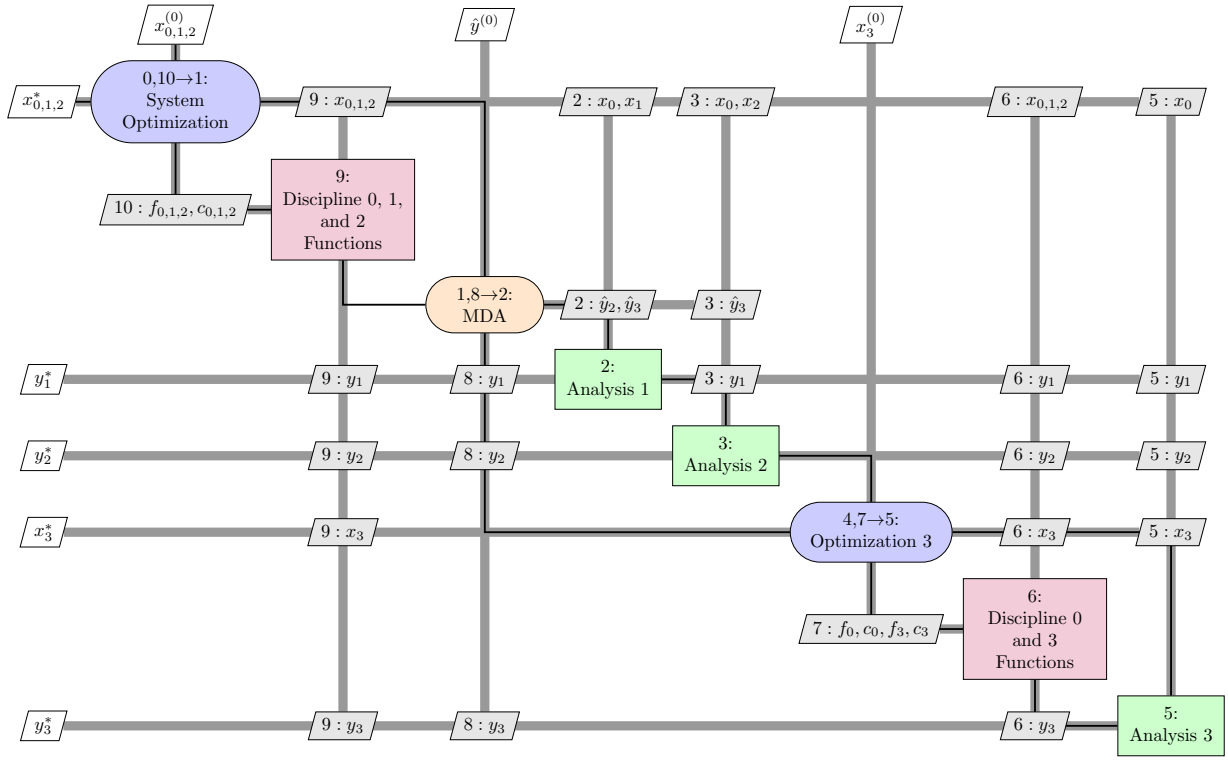

Figure 17. Diagram for the ASO architecture.

Algorithm 11 Asymmetric subspace optimization (ASO)

Input: Initial design variables $x^{(0)}$

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 $7 \rightarrow 5$: Discipline 3 optimization has converged
    8: Update coupling variables
  until $8 \rightarrow 2$ MDA has converged
  9: Compute objective and constraint function values for disciplines 1 and 2
  10: Update design variables
until $10 \rightarrow 1$: 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 [120], to calculate gradients at the system level. CPOS represents an extension of the coupled sensitivity equations [107, 98] to include the optimality conditions. ASO was later implemented using a coupled-adjoint approach as well [75]. Kennedy et al. [196] present alternative strategies for computing the discipline 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. We also note the work of Kaufman et al. [197] and Hosder et al. [198], who use structural optimization as an inner loop in aircraft design optimization with a surrogate model replacing the post-optimality sensitivity equations.

The system subproblem is

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\left(x, y\right)\right) \\ \text{with respect to} \quad & x_0, x_k \\ \text{subject to} \quad & c_0\left(x, y\left(x, y\right)\right) \geq 0 \\ & c_k\left(x_0, x_k, y_k\left(x_0, x_k, y_{j\neq k}\right)\right) \geq 0 \quad \text{for all } k, \end{aligned} \tag{25}$$

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

$$\begin{aligned} \text{minimize} \quad & f_0\left(x, y\left(x, y\right)\right) \\ \text{with respect to} \quad & x_i \\ \text{subject to} \quad & c_i\left(x_0, x_i, y_i\left(x_0, x_i, y_{j\neq i}\right)\right) \geq 0. \end{aligned} \tag{26}$$

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 [120]. However, the total time for the optimization routine is

competitive with MDF only if the aerodynamic analysis is substantially more costly than the structural analysis. If the two analyses take roughly the same time, MDF is still much faster. Furthermore, using CPOS increases the complexity of the sensitivity analysis compared to using a normal coupled adjoint, which adds to the overall solution time. As a result, this architecture may appeal only to practitioners solving MDO problems with widely varying computational costs in the discipline analysis.

V. Architecture Benchmarking Issues

One of the challenges facing the MDO community is determining which architecture is most efficient for a given MDO problem or class of problems. One way is to benchmark 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, since the most appropriate architecture often depends on the nature of the problem. There are a number of benchmarking studies in the literature [156, 157, 199, 134, 143, 200, 135, 136]. However, they have limitations that need to be considered when reading these studies and considering future work.

The first is the fact that no two studies are likely to produce identical results. This is because no two programmers will program the same architecture in the same way. In addition, experts in a particular architecture may be able to more fully optimize the performance of that architecture, unintentionally biasing the results [135]. This problem can be overcome by performing the benchmarking in a specialized MDO framework such as iSight, ModelCenter, pyMDO [201], or OpenMDAO [202]. 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. OpenMDAO and pyMDO can even implement the MDO architectures automatically [201]. A framework has also been proposed to go a step further and select architectures automatically based on the structure of the problem and nest different architecture in the solution of large MDO problems [203]. Padula and Gillian [204] present a brief history of frameworks for MDO. Kodiyalam and Sobieski [205] and Kodiyalam et al. [206] discuss requirements for an MDO framework and high-performance computing considerations respectively.

The second limitation is the choice of architectures for 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 as the newer architectures become better known. However, we must emphasize that implementing a new architecture in an established MDO framework should allow more rapid benchmarking and give a much earlier indication of the architecture's promise.

The third limitation is in the test problems themselves. Almost all 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 with 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 should 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 [136] 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. [207]. Test problems with lower dimensionality but with multiple local minima are presented by Tosserams et al. [208]. We also feel that a test suite of MDO problems similar to the now-defunct NASA MDO suite [209] would be a great resource for researchers and practitioners.

VI. Conclusions and Future Work

In this paper, we summarized and classified all known MDO architectures. We presented the architectures in a unified notation to enable comparisons of them, thus contributing to a deeper and more encompassing understanding of the theory. We also introduced a classification of the architectures—summarized in Fig. 7—that built on previous work to show how some architectures may be derived from others. Our hope is that this work will help point the way toward new MDO architectures and expand the scope of MDO theory and applications.

From a more practical perspective, much work remains to be done in the area of benchmarking the existing architectures. The performance of the new architectures needs to be compared with that of the older architectures on a predefined set of test problems, and the results should be independently corroborated. The development of MDO frameworks and test problem suites would be extremely useful.

Finally, new architectures are needed. A distributed architecture that exhibits fast convergence in most medium- and large-scale problems may be thought of as the "holy grail" of MDO. The newest architectures in this survey represent significant progress toward this objective. However, the distributed architectures still seem to be more expensive than the monolithic architectures they are meant to supersede. Until a distributed architecture can demonstrate fast convergence for a wide range of problems, architecture development will remain an active area of research.

VII. Acknowledgments

We would like to thank the following colleagues for providing feedback on a preprint of this paper: Evin Cramer, Pascal Etman, Raphael Haftka, John Hwang, Harrison Kim, Brian Roth, and Jaroslaw Sobieski. Additional helpful suggestions were provided by James Allison, Sylvain Arreckx, Cristina Bloebaum, Robert Canfield, Justin Gray, Jason Hicken, Graeme Kennedy, Alicia Kim, Michael Kokkolaras, Zhoujie Lyu, and Kenneth Moore. We are also grateful to the anonymous reviewers for their diligence and insights. This work was partially funded by a scholarship from the Natural Science and Engineering Research Council of Canada.

References

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

[2] Schmit, L. A. and Thornton, W. A., "Synthesis of an Airfoil at Supersonic Mach Number," Tech. Rep. CR 144, NASA, Jan. 1965.

[3] Schmit, L. A., "Structural Synthesis — Its Genesis and Development," *AIAA Journal*, Vol. 19, No. 10, 1981, pp. 1249–1263. doi:10.2514/3.7859.

[4] Schmit Jr., L. A., "Structural Synthesis — Precursor and Catalyst. Recent Experiences in Multidisciplinary Analysis and Optimization," Tech. Rep. CP-2337, NASA, 1984.

[5] Haftka, R. T., "Automated Procedure for Design of Wing Structures to Satisfy Strength and Flutter Requirements," Tech. Rep. TN D-7264, NASA Langley Research Center, Hampton, VA, 1973.

[6] Haftka, R. T., Starnes Jr., J. H., Barton, F. W., and Dixon, S. C., "Comparison of Two Types of Optimization Procedures for Flutter Requirements," *AIAA Journal*, Vol. 13, No. 10, 1975, pp. 1333–1339. doi:10.2514/3.60545.

[7] Haftka, R. T., "Optimization of Flexible Wing Structures Subject to Strength and Induced Drag Constraints," *AIAA Journal*, Vol. 14, No. 8, 1977, pp. 1106–1977. doi:10.2514/3.7400.

[8] Haftka, R. T. and Shore, C. P., "Approximate Methods for Combined Thermal/Structural Design," Tech. Rep. TP-1428, NASA, June 1979.

[9] Ashley, H., "On Making Things the Best — Aeronautical Uses of Optimization," *Journal of Aircraft*, Vol. 19, No. 1, 1982, pp. 5–28.

[10] Green, J. A., "Aeroelastic tailoring of aft-swept high-aspect-ratio composite wings," *Journal of Aircraft*, Vol. 24, No. 11, Nov. 1987, pp. 812–819. doi:10.2514/3.45525.

[11] Grossman, B., Gurdal, Z., Strauch, G. J., Eppard, W. M., and Haftka, R. T., "Integrated Aerodynamic/Structural Design of a Sailplane Wing," *Journal of Aircraft*, Vol. 25, No. 9, 1988, pp. 855–860. doi:10.2514/3.45670.

[12] Grossman, B., Haftka, R. T., Kao, P.-J., Polen, D. M., and Rais-Rohani, M., "Integrated Aerodynamic-Structural Design of a Transport Wing," *Journal of Aircraft*, Vol. 27, No. 12, 1990, pp. 1050–1056. doi:10.2514/3.45980.

[13] Livne, E., Schmit, L., and Friedmann, P., "Towards Integrated Multidisciplinary Synthesis of Actively Controlled Fiber Composite Wings," *Journal of Aircraft*, Vol. 27, No. 12, Dec. 1990, pp. 979–992. doi:10.2514/3.45972.

[14] Livne, E., "Integrated Aeroservoelastic Optimization: Status and Direction," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 122–145. doi:10.2514/2.2419.

[15] Jansen, P., Perez, R. E., and Martins, J. R. R. A., "Aerostructural Optimization of Nonplanar Lifting Surfaces," *Journal of Aircraft*, Vol. 47, No. 5, 2010, pp. 1491–1503. doi:10.2514/1.44727.

[16] Ning, S. A. and Kroo, I., "Multidisciplinary Considerations in the Design of Wings and Wing Tip Devices," *Journal of Aircraft*, Vol. 47, No. 2, March 2010, pp. 534–543. doi:10.2514/1.41833.

[17] Kroo, I. M., Altus, S., Braun, R., Gage, P., and Sobieski, I., "Multidisciplinary Optimization Methods for Aircraft Preliminary Design," *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1994.

[18] Manning, V. M., *Large-Scale Design of Supersonic Aircraft via Collaborative Optimization*, Ph.D. thesis, Stanford University, 1999.

[19] Antoine, N. E. and Kroo, I. M., "Framework for Aircraft Conceptual Design and Environmental Performance Studies," *AIAA Journal*, Vol. 43, No. 10, Oct. 2005, pp. 2100–2109.

[20] Henderson, R. P., Martins, J. R. R. A., and Perez, R. E., "Aircraft Conceptual Design for Optimal Environmental Performance," *The Aeronautical Journal*, Vol. 116, No. 1175, Jan. 2012, pp. 1–22.

[21] Alonso, J. J. and Colonno, M. R., "Multidisciplinary Optimization with Applications to Sonic-Boom Minimization," *Annual Review of Fluid Mechanics*, Vol. 44, No. 1, 2012, pp. 505–526. doi:10.1146/annurev-fluid-120710-101133.

[22] Balling, R. and Rawlings, M. R., "Collaborative Optimization with Disciplinary Conceptual Design," *Structural and Multidisciplinary Optimization*, Vol. 20, No. 3, 2000, pp. 232–241. doi:10.1007/s001580050151.

[23] Choudhary, R., Malkawi, A., and Papalambros, P. Y., "Analytic Target Cascading in Simulation-Based Building Design," *Automation in Construction*, Vol. 14, No. 4, 2005, pp. 551–568. doi:10.1016/j.autcon.2004.11.004.

[24] Geyer, P., "Component-Oriented Decomposition for Multidisciplinary Design Optimization in Building Design," *Advanced Engineering Informatics*, Vol. 23, No. 1, 2009, pp. 12–31. doi:10.1016/j.aei.2008.06.008.

[25] He, Y. and McPhee, J., "Multidisciplinary Optimization of Multibody Systems with Application to the Design of Rail Vehicles," *Multibody System Dynamics*, Vol. 14, No. 2, 2005, pp. 111–135. doi:10.1007/s11044-005-4310-0.

[26] Enblom, R., "Two-Level Numerical Optimization of Ride Comfort in Railway Vehicles," *Journal of Rail and Rapid Transit*, Vol. 220, No. 1, 2006, pp. 1–11. doi:10.1243/095440905X33279.

[27] Potsaid, B., Bellouard, Y., and Wen, J. T.-Y., "A Multidisciplinary Design and Optimization Methodology for the Adaptive Scanning Optical Microscope (ASOM)," *Proceedings of the SPIE*, Vol. 6289, 2006, pp. 62890L1–12. doi:10.1117/12.680450.

[28] McAllister, C. D. and Simpson, T. W., "Multidisciplinary Robust Design Optimization of an Internal Combustion Engine," *Journal of Mechanical Design*, Vol. 125, No. 1, 2003, pp. 124–130. doi:10.1115/1.1543978.

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

[30] Peri, D. and Campana, E. F., "Multidisciplinary Design Optimization of a Naval Surface Combatant," *Journal of Ship Research*, Vol. 47, No. 1, 2003, pp. 1–12.

[31] Kalavalapally, R., Penmetsa, R., and Grandhi, R., "Multidisciplinary optimization of a lightweight torpedo structure subjected to an underwater explosion," *Finite Elements in Analysis and Design*, Vol. 43, No. 2, Dec. 2006, pp. 103–111. doi:10.1016/j.finel.2006.07.005.

[32] Takekoshi, Y., Kawamura, T., Yamaguchi, H., Maeda, M., Ishii, N., Kimura, K., Taketani, T., and Fujii, A., "Study on the design of propeller blade sections using the optimization algorithm," *Journal of Marine Science and Technology*, Vol. 10, 2005, pp. 70–81. doi:10.1007/s00773-005-0197-y.

[33] Young, Y., Baker, J., and Motley, M., "Reliability-based design and optimization of adaptive marine structures," *Composite Structures*, Vol. 92, 2010, pp. 244–253. doi:10.1016/j.compstruct.2009.07.024.

[34] Ganguli, R., "Survey of Recent Developments in Rotorcraft Design Optimization," *Journal of Aircraft*, Vol. 41, No. 3, 2004, pp. 493–510. doi:10.2514/1.58.

[35] Glaz, B., Friedmann, P. P., and Liu, L., "Helicopter Vibration Reduction Throughout the Entire Flight Envelope Using Surrogate-Based Optimization," *Journal of the American Helicopter Society*, Vol. 54, 2009, pp. 12007—-1–15. doi:10.4050/JAHS.54.012007.

[36] Fuglsang, P. and Madsen, H., "Optimization Method for Wind Turbine Rotors," *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 80, 1999, pp. 191–206.

[37] Fuglsang, P., Bak, C., Schepers, J. G., Bulder, B., Cockerill, T. T., Claiden, P., Olesen, A., and van Rossen, R., "Site-Specific Design Optimization of Wind Turbines," *Wind Energy*, Vol. 5, 2002, pp. 261–279.

[38] Kenway, G. and Martins, J. R. R. A., "Aerostructural Shape Optimization of Wind Turbine Blades Considering Site-Specific Winds," *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, BC, Sept. 2008, AIAA 2008-6025.

[39] Braun, R. D., Moore, A. A., and Kroo, I. M., "Collaborative Approach to Launch Vehicle Design," *Journal of Spacecraft and Rockets*, Vol. 34, No. 4, 1997, pp. 478–486. doi:10.2514/2.3237.

[40] Cai, G., Fang, J., Zheng, Y., Tong, X., Chen, J., and Wang, J., "Optimization of System Parameters for Liquid Rocket Engines with Gas-Generator Cycles," *Journal of Propulsion and Power*, Vol. 26, No. 1, 2010, pp. 113–119. doi:10.2514/1.40649.

[41] Zingg, D. W., Nemec, M., and Pulliam, T. H., "A Comparative Evaluation of Genetic and Gradient-Based Algorithms Applied to Aerodynamic Optimization," *Shape design in aerodynamics REMN – 17/2008.*, 2008, pp. 103–126.

[42] Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer-Verlag, 2nd ed., 2006.

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

[44] Lewis, K. and Mistree, F., "The Other Side of Multidisciplinary Design Optimization: Accommodating a Multiobjective, Uncertain and Non-Deterministic World," *Engineering Optimization*, Vol. 31, 1998, pp. 161–189. doi:10.1080/03052159808941369.

[45] Miettinen, K. M., *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, 1998.

[46] Haftka, R. T. and Watson, L. T., "Decomposition Theory for Multidisciplinary Design Optimization Problems with Mixed Integer Quasiseparable Subsystems," *Optimization and Engineering*, Vol. 7, No. 2, 2006, pp. 135–149. doi:10.1007/s11081-006-6836-2.

[47] Michalek, J. J. and Papalambros, P. Y., "BB-ATC: Analytical Target Cascading Using Branch and Bound for Mixed Integer Nonlinear Programming," *Proceedings of the ASME Design Engineering Technical Conference*, 2006.

[48] Zadeh, P. M., Toropov, V. V., and Wood, A. S., "Metamodel-Based Collaborative Optimization Framework," *Structural and Multidisciplinary Optimization*, Vol. 38, 2009, pp. 103–115. doi:10.1007/s00158-008-0286-8.

[49] Conn, A. R., Schienburg, K., and Vicente, L. N., *Introduction to Derivative-Free Optimization*, SIAM, 2009.

[50] Rao, S. S., "Game Theory Approach for Multiobjective Structural Optimization," *Computers and Structures*, Vol. 25, No. 1, 1987, pp. 119–127. doi:10.1016/0045-7949(87)90223-9.

[51] Lewis, K. and Mistree, F., "Modeling Interactions in Multidisciplinary Design: A Game Theoretic Approach," *AIAA Journal*, Vol. 35, No. 8, 1997, pp. 1387–1392. doi:10.2514/2.248.

[52] Honda, T., Cucci, F., and Yang, M. C., "Achieving Pareto Optimality in a Decentralized Design Environment," *International Conference on Engineering Design*, Stanford, CA, 2009.

[53] Sobieszczanski-Sobieski, J., Altus, T. D., Phillips, M., and Sandusky Jr., R. R., "Bilevel Integrated System Synthesis for Concurrent and Distributed Processing," *AIAA Journal*, Vol. 41, No. 10, 2003, pp. 1996–2003. doi:10.2514/2.1889.

[54] Tosserams, S., Kokkolaras, M., Etman, L. F. P., and Rooda, J. E., "A Nonhierarchical Formulation of Analytical Target Cascading," *Journal of Mechanical Design*, Vol. 132, No. 5, 2010, pp. 051002—-1–13. doi:10.1115/1.4001346.

[55] Roth, B. and Kroo, I., "Enhanced Collaborative Optimization: Application to an Analytic Test Problem and Aircraft Design," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, BC, Canada, Sept. 2008, AIAA 2008-5841.

[56] Agte, J., de Weck, O., Sobieszczanski-Sobieski, J., Arendsen, P., Morris, A., and Spieck, M., "MDO: Assessment and Direction for Advancement — An Opinion of One International Group," *Structural and Multidisciplinary Optimization*, Vol. 40, 2010, pp. 17–33. doi:10.1007/s00158-009-0381-5.

[57] Kim, H. M., Michelena, N. F., Papalambros, P. Y., and Jiang, T., "Target Cascading in Optimal System Design," *Journal of Mechanical Design*, Vol. 125, No. 3, 2003, pp. 474–480. doi:10.1115/1.1582501.

[58] Michelena, N. F., Park, H., and Papalambros, P. Y., "Convergence Properties of Analytical Target Cascading," *AIAA Journal*, Vol. 41, No. 5, 2003, pp. 897–905. doi:10.2514/2.2025.

[59] Michalek, J. J. and Papalambros, P. Y., "An Efficient Weighting Update Method to Achieve Acceptable Consistency Deviation in Analytical Target Cascading," *Journal of Mechanical Design*, Vol. 127, 2005, pp. 206–214. doi:10.1115/1.1830046.

[60] Cramer, E. J., Dennis Jr., J. E., Frank, P. D., Lewis, R. M., and Shubin, G. R., "Problem Formulation for Multidisciplinary Optimization," *SIAM Journal on Optimization*, Vol. 4, No. 4, 1994, pp. 754–776. doi:10.1137/0804044.

[61] Alexandrov, N. M. and Lewis, R. M., "Analytical and Computational Aspects of Collaborative Optimization for Multidisciplinary Design," *AIAA Journal*, Vol. 40, No. 2, 2002, pp. 301–309. doi:10.2514/2.1646.

[62] Alexandrov, N. M. and Lewis, R. M., "Reconfigurability in MDO Problem Synthesis, Part 1," *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, Sept. 2004, AIAA 2004-4307.

[63] Alexandrov, N. M. and Lewis, R. M., "Reconfigurability in MDO Problem Synthesis, Part 2," *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, Sept. 2004, AIAA 2004-4308.

[64] Wujek, B. A., Renaud, J. E., Batill, S. M., and Brockman, J. B., "Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment," *Concurrent Engineering: Research and Applications*, Vol. 4, No. 4, 1996, pp. 361–377. doi:10.1177/1063293X9600400405.

[65] Haftka, R. T. and Watson, L. T., "Multidisciplinary Design Optimization with Quasiseparable Subsystems," *Optimization and Engineering*, Vol. 6, 2005, pp. 9–20. doi:10.1023/B:OPTE.0000048534.58121.93.

[66] Sobieszczanski-Sobieski, J. and Haftka, R. T., "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments," *Structural Optimization*, Vol. 14, No. 1, 1997, pp. 1–23. doi:10.1007/BF01197554.

[67] Kodiyalam, S. and Sobieszczanski-Sobieski, J., "Bilevel Integrated System Synthesis with Response Surfaces," *AIAA Journal*, Vol. 38, No. 8, 2000, pp. 1479–1485. doi:10.2514/2.1126.

[68] Sellar, R. S., Batill, S. M., and Renaud, J. E., "Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design," *Proceedings of the 34th AIAA Aerospace Sciences and Meeting Exhibit*, Reno, NV, Jan. 1996, AIAA 1996-0714.

[69] Sobieszczanski-Sobieski, J., Agte, J. S., and Sandusky Jr., R. R., "Bilevel Integrated System Synthesis," *AIAA Journal*, Vol. 38, No. 1, 2000, pp. 164–172. doi:10.1.1.35.4601.

[70] Shin, M.-K. and Park, G.-J., "Multidisciplinary Design Optimization Based on Independent Subspaces," *International Journal for Numerical Methods in Engineering*, Vol. 64, 2005, pp. 599–617. doi:10.1002/nme.1380.

[71] DeMiguel, V. and Murray, W., "A Local Convergence Analysis of Bilevel Decomposition Algorithms," *Optimization and Engineering*, Vol. 7, No. 2, 2006, pp. 99–133. doi:10.1007/s11081-006-6835-3.

[72] Braun, R. D. and Kroo, I. M., "Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. Alexandrov and M. Y. Hussaini, SIAM, 1997, pp. 98–116.

[73] Kroo, I. M., "MDO for Large-Scale Design," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. Alexandrov and M. Y. Hussaini, SIAM, 1997, pp. 22–44.

[74] Sobieski, I. P. and Kroo, I. M., "Collaborative Optimization Using Response Surface Estimation," *AIAA Journal*, Vol. 38, No. 10, 2000, pp. 1931–1938. doi:10.2514/2.847.

[75] Chittick, I. R. and Martins, J. R. R. A., "Aero-Structural Optimization Using Adjoint Coupled Post-Optimality Sensitivities," *Structural and Multidisciplinary Optimization*, Vol. 36, 2008, pp. 59–70. doi:10.1007/s00158-007-0200-9.

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

[77] Balling, R. J. and Sobieszczanski-Sobieski, J., "Optimization of Coupled Systems: A Critical Overview of Approaches," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 6–17. doi:10.2514/3.13015.

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

[79] Alexandrov, N. M., "Multilevel Methods for MDO," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. M. Alexandrov and M. Y. Hussaini, SIAM, Philadelphia, 1997.

[80] Balling, R. J., "Approaches to MDO Which Support Disciplinary Autonomy," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. M. Alexandrov and M. Y. Hussaini, SIAM, 1997, pp. 90–97.

[81] Willcox, K. and Wakayama, S., "Simultaneous Optimization of a Multiple-Aircraft Family," *Journal of Aircraft*, Vol. 40, No. 4, July 2003, pp. 616–622. doi:10.2514/2.3156.

[82] Piperni, P., Abdo, M., Kafyeke, F., and Isikveren, A. T., "Preliminary Aerostructural Optimization of a Large Business Jet," *Journal of Aircraft*, Vol. 44, No. 5, 2007, pp. 1422–1438.

[83] Simpson, T. W. and Martins, J. R. R. A., "Multidisciplinary Design Optimization for Complex Engineered Systems Design: Report from an NSF Workshop," *Journal of Mechanical Design*, Vol. 133, No. 10, Oct. 2011, pp. 101002. doi:10.1115/1.4004465.

[84] Tosserams, S., Etman, L. F. P., and Rooda, J. E., "A Classification of Methods for Distributed System Optimization based on Formulation Structure," *Structural and Multidisciplinary Optimization*, Vol. 39, No. 5, 2009, pp. 503–517. doi:10.1007/s00158-008-0347-z.

[85] Lambe, A. B. and Martins, J. R. R. A., "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, Vol. 46, 2012, pp. 273–284. doi:10.1007/s00158-012-0763-y.

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

[87] Browning, T. R., "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions on Engineering Management*, Vol. 48, No. 3, 2001, pp. 292–306. doi:10.1109/17.946528.

[88] Haftka, R. T., "Simultaneous Analysis and Design," *AIAA Journal*, Vol. 23, No. 7, 1985, pp. 1099–1103. doi:10.2514/3.9043.

[89] Schmit Jr., L. A. and Ramanathan, R. K., "Multilevel Approach to Minimum Weight Design Including Buckling Constraints," *AIAA Journal*, Vol. 16, No. 2, 1978, pp. 97–104. doi:10.2514/3.60867.

[90] Biegler, L. T., Ghattas, O., Heinkenschloss, M., and van Bloemen Waanders, B., editors, *Large-Scale PDE-Constrained Optimization*, Springer-Verlag, 2003.

[91] Squire, W. and Trapp, G., "Using Complex Variables to Estimate Derivatives of Real Functions," *SIAM Review*, Vol. 40, No. 1, 1998, pp. 110–112.

[92] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, 2003, pp. 245–262. doi:10.1145/838250.838251.

[93] Adelman, H. M. and Haftka, R. T., "Sensitivity Analysis of Discrete Structural Systems," *AIAA Journal*, Vol. 24, No. 5, 1986, pp. 823–832. doi:10.2514/3.48671.

[94] Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, Sept. 1988, pp. 233–260.

[95] Giles, M. B., Duta, M. C., Müller, J.-D., and Pierce, N. A., "Algorithm Developments for Discrete Adjoint Methods," *AIAA Journal*, Vol. 41, No. 2, Feb. 2003, pp. 198–205.

[96] van Keulen, F., Haftka, R., and Kim, N., "Review of options for structural design sensitivity analysis. Part 1: Linear systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, 2005, pp. 3213–3243.

[97] Mader, C. A., Martins, J. R. R. A., Alonso, J. J., and van der Weide, E., "ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers," *AIAA Journal*, Vol. 46, No. 4, April 2008, pp. 863–873. doi:10.2514/1.29123.

[98] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "A Coupled-Adjoint Sensitivity Analysis Method for High-Fidelity Aero-Structural Design," *Optimization and Engineering*, Vol. 6, No. 1, March 2005, pp. 33–62. doi:10.1023/B:OPTE.0000048536.47956.62.

[99] Barcelos, M., Bavestrello, H., and Maute, K., "A Schur–Newton–Krylov solver for steady-state aeroelastic analysis and design sensitivity analysis," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, 2006, pp. 2050—2069.

[100] Hicken, J. and Zingg, D., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," *AIAA Journal*, Vol. 48, No. 2, Feb. 2009, pp. 400–413.

[101] Martins, J. R. R. A. and Hwang, J. T., "A Unification of Discrete Methods for Computing Derivatives of Single- and Multi-disciplinary Computational Models," *AIAA Journal*, 2012, (Accepted subject to revisions).

[102] Bloebaum, C., "Coupling strength-based system reduction for complex engineering design," *Structural Optimization*, Vol. 10, 1995, pp. 113–121.

[103] Kennedy, G. J. and Martins, J. R. R. A., "Parallel Solution Methods for Aerostructural Analysis and Design Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Forth Worth, TX, Sept. 2010, AIAA 2010-9308.

[104] Vanderplaats, G. N., "An Efficient Feasible Directions Algorithm for Design Synthesis," *AIAA Journal*, Vol. 22, No. 11, 1984, pp. 1633–1640. doi:10.2514/3.8829.

[105] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, Vol. 47, No. 1, 2005, pp. 99–131. doi:10.1137/S0036144504446096.

[106] Wright, S. J., *Primal-Dual Interior Point Methods*, SIAM, 1997.

[107] Sobieszczanski-Sobieski, J., "Sensitivity of Complex, Internally Coupled Systems," *AIAA Journal*, Vol. 28, No. 1, 1990, pp. 153–160. doi:10.2514/3.10366.

[108] Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "A Scalable Parallel Approach for High-Fidelity Aerostructural Analysis and Optimization," *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Honolulu, HI, April 2012, AIAA 2012-1922.

[109] Marriage, C. J. and Martins, J. R. R. A., "Reconfigurable Semi-Analytic Sensitivity Methods and MDO Architectures Within the πMDO Framework," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, BC, Canada, Sept. 2008, AIAA 2008-5956.

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

[111] Conejo, A. J., Nogales, F. J., and Prieto, F. J., "A Decomposition Procedure Based on Approximate Newton Directions," *Mathematical Programming*, Vol. 93, 2002, pp. 495–515. doi:10.1007/s10107-002-0304-3.

[112] Dantzig, G. B. and Wolfe, P., "Decomposition Principle for Linear Programs," *Operations Research*, Vol. 8, 1960, pp. 101–111. doi:10.1287/opre.8.1.101.

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

[114] Michelena, N. F. and Papalambros, P. Y., "A Network Reliability Approach to Optimal Decomposition of Design Problems," *Journal of Mechanical Design*, Vol. 2, 1995, pp. 195–204. doi:10.1.1.35.147.

[115] Michelena, N. F. and Papalambros, P. Y., "A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems," *Computational Optimization and Applications*, Vol. 8, 1997, pp. 173–196. doi:10.1023/A:1008673321406.

[116] Tosserams, S., Hofkamp, A. T., Etman, L. F. P., and Rooda, J. E., "A Specification Language for Problem Partitioning in Decomposition-Based Design Optimization," *Structural and Multidisciplinary Optimization*, Vol. 42, 2010, pp. 707–723. doi:10.1007/s00158-010-0512-z.

[117] Bertsekas, D. P. and Tsitsiklis, J. N., *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1997.

[118] Zadeh, P. M. and Toropov, V. V., "Multi-Fidelity Multidisciplinary Design Optimization Based on Collaborative Optimization Framework," *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, Sept. 2002, AIAA-2002-5504.

[119] Robinson, T. D., Willcox, K. E., Eldred, M. S., and Haimes, R., "Multifidelity Optimization for Variable Complexity Design," *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, VA, Sept. 2006, AIAA 2006-7114.

[120] Chittick, I. R. and Martins, J. R. R. A., "An Asymmetric Suboptimization Approach to Aerostructural Optimization," *Optimization and Engineering*, Vol. 10, No. 1, March 2009, pp. 133–152. doi:10.1007/s11081-008-9046-2.

[121] Alexandrov, N. M. and Lewis, R. M., "Comparative Properties of Collaborative Optimization and Other Approaches to MDO," *1st ASMO UK/ISSMO Conference on Engineering Design Optimization*, 1999.

[122] De Wit, A. J. and Van Keulen, F., "Overview of Methods for Multilevel and/or Multidisciplinary Optimization," *Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Orlando, FL, April 2010, AIAA 2010-2914.

[123] Geethaikrishnan, C., Mujumdar, P. M., Sudhakar, K., and Adimurthy, V., "A Hybrid MDO Architecture for Launch Vehicle Conceptual Design," *Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Orlando, FL, April 2010, AIAA 2010-2757.

[124] Sobieszczanski-Sobieski, J., "Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems," Tech. rep., NASA Langley Research Center, Hampton, VA, Sept. 1988.

[125] Bloebaum, C. L., Hajela, P., and Sobieszczanski-Sobieski, J., "Non-Hierarchic System Decomposition in Structural Optimization," *Engineering Optimization*, Vol. 19, No. 3, 1992, pp. 171–186. doi:10.1080/03052159208941227.

[126] Shankar, J., Ribbens, C. J., Haftka, R. T., and Watson, L. T., "Computational Study of a Nonhierarchical Decomposition Algorithm," *Computational Optimization and Applications*, Vol. 2, 1993, pp. 273–293. doi:10.1007/BF01299452.

[127] Renaud, J. E. and Gabriele, G. A., "Approximation in Non-Hierarchic System Optimization," *AIAA Journal*, Vol. 32, No. 1, 1994, pp. 198–205. doi:10.2514/3.11967.

[128] Renaud, J. E. and Gabriele, G. A., "Improved Coordination in Nonhierarchic System Optimization," *AIAA Journal*, Vol. 31, No. 12, 1993, pp. 2367–2373. doi:10.2514/3.11938.

[129] Parashar, S. and Bloebaum, C. L., "Multi-Objective Genetic Algorithm Concurrent Subspace Optimization (MOGACSSO) for Multidisciplinary Design," *In Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Newport, RI, May 2006, AIAA 2006-2047.

[130] Huang, C.-H., Galuski, J., and Bloebaum, C. L., "Multi-Objective Pareto Concurrent Subspace Optimization for Multidisciplinary Design," *AIAA Journal*, Vol. 45, No. 8, 2007, pp. 1894–1906. doi:10.2514/1.19972.

[131] Zhang, K.-S., Han, Z.-H., Li, W.-J., and Song, W.-P., "Bilevel Adaptive Weighted Sum Method for Multidisciplinary Multi-Objective Optimization," *AIAA Journal*, Vol. 46, No. 10, 2008, pp. 2611–2622. doi:10.2514/1.36853.

[132] Parashar, S. and Bloebaum, C. L., "Robust Multi-Objective Genetic Algorithm Concurrent Subspace Optimization (R-MOGACSSO) for Multidisciplinary Design," *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Porthsmouth, VA, Sept. 2006, AIAA 2006-6907.

[133] Tappeta, R. V., Nagendra, S., and Renaud, J. E., "A Multidisciplinary Design Optimization Approach for High Temperature Aircraft Engine Components," *Structural Optimization*, Vol. 18, No. 2-3, 1999, pp. 134–145. doi:10.1007/BF01195988.

[134] Perez, R. E., Liu, H. H. T., and Behdinan, K., "Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design," *Proceedings of the 10th AIAA/ISSMO Muldisiciplinary Analysis and Optimization Conference*, Albany, NY, Aug. 2004, AIAA 2004-4537.

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

[136] Tedford, N. P. and Martins, J. R. R. A., "Benchmarking Multidisciplinary Design Optimization Algorithms," *Optimization and Engineering*, Vol. 11, 2010, pp. 159–183. doi:10.1007/s11081-009-9082-6.

[137] de Wit, A. J. and van Keulen, F., "Numerical Comparison of Multi-level Optimization Techniques," *Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Honolulu, HI, April 2007, AIAA 2007-1895.

[138] Braun, R. D., *Collaborative Optimization: An Architecture for Large-Scale Distributed Design*, Ph.D. thesis, Stanford University, Stanford, CA, 1996.

[139] Braun, R. D., Gage, P. J., Kroo, I. M., and Sobieski, I. P., "Implementation and Performance Issues in Collaborative Optimization," *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Multidisciplinary Analysis and Optimization Symposium*, Bellevue, WA, Sept. 1996, AIAA 1996-4017.

[140] DeMiguel, A.-V. and Murray, W., "An Analysis of Collaborative Optimization Methods," *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, 2000, AIAA 2000-4720.

[141] Lin, J. G., "Analysis and Enhancement of Collaborative Optimization for Multidisciplinary Design," *AIAA Journal*, Vol. 42, No. 2, 2004, pp. 348–360. doi:10.2514/1.9098.

[142] Fiacco, A. V. and McCormick, G. P., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, SIAM, 1990.

[143] Brown, N. F. and Olds, J. R., "Evaluation of Multidisciplinary Optimization Techniques Applied to a Reusable Launch Vehicle," *Journal of Spacecraft and Rockets*, Vol. 43, No. 6, 2006, pp. 1289–1300. doi:10.2514/1.16577.

[144] Perez, R. E., *A Multidisciplinary Optimization Framework for Flight Dynamics and Control Integration in Aircraft Design*, Ph.D. thesis, University of Toronto, 2007.

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

[146] Li, X., Li, W., and Liu, C., "Geometric Analysis of Collaborative Optimization," *Structural and Multidisciplinary Optimization*, Vol. 35, 2008, pp. 301–313. doi:10.1007/s00158-007-0127-1.

[147] Budianto, I. A. and Olds, J. R., "Design and Deployment of a Satellite Constellation Using Collaborative Optimization," *Journal of Spacecraft and Rockets*, Vol. 41, No. 6, 2004, pp. 956–963. doi:10.2514/1.14254.

[148] Ledsinger, L. A. and Olds, J. R., "Optimized Solutions for Kistler K-1 Branching Trajectories Using Multidisciplinary Design Optimization Techniques," *Journal of Spacecraft and Rockets*, Vol. 39, No. 3, 2002, pp. 420–429. doi:10.2514/2.3825.

[149] Perez, R. E., Liu, H. H. T., and Behdinan, K., "Multidisciplinary Optimization Framework for Control-Configuration Integration in Aircraft Conceptual Design," *Journal of Aircraft*, Vol. 43, No. 6, 2006, pp. 1937–1948. doi:10.2514/1.22263.

[150] Allison, J. T., Roth, B., Kokkolaras, M., Kroo, I. M., and Papalambros, P. Y., "Aircraft Family Design Using Decomposition-Based Methods," *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Porthsmouth, VA, Sept. 2006, AIAA 2006-6950.

[151] McAllister, C. D., Simpson, T. W., Hacker, K., Lewis, K., and Messac, A., "Integrating Linear Physical Programming Within Collaborative Optimization for Multiobjective Multidisciplinary Design Optimization," *Structural and Multidisciplinary Optimization*, Vol. 29, No. 3, 2005, pp. 178–189. doi:10.1007/s00158-004-0481-1.

[152] Gu, X., Renaud, J. E., Ashe, L. M., Batill, S. M., Budhiraja, A. S., and Krajewski, L. J., "Decision-Based Collaborative Optimization," *Journal of Mechanical Design*, Vol. 124, No. 1, 2002, pp. 1–13. doi:10.1115/1.1432991.

[153] Gu, X. S., Renaud, J. E., and Penninger, C. L., "Implicit Uncertainty Propagation for Robust Collaborative Optimization," *Journal of Mechanical Design*, Vol. 128, No. 4, 2006, pp. 1001–1013. doi:10.1115/1.2205869.

[154] Huang, H.-Z., Tao, Y., and Liu, Y., "Multidisciplinary Collaborative Optimization Using Fuzzy Satisfaction Degree and Fuzzy Sufficiency Degree Model," *Soft Computing*, Vol. 12, 2008, pp. 995–1005. doi:10.1007/s00500-007-0268-6.

[155] Roth, B. D., *Aircraft Family Design Using Enhanced Collaborative Optimization*, Ph.D. thesis, Stanford University, 2008.

[156] Kodiyalam, S., "Evaluation of Methods for Multidisciplinary Design Optimization (MDO), Phase I," Tech. Rep. CR-1998-208716, NASA, Sept. 1998.

[157] Kodiyalam, S. and Yuan, C., "Evaluation of Methods for Multidisciplinary Design Optimization (MDO), Part 2," Tech. Rep. CR-2000-210313, NASA, Nov. 2000.

[158] Conn, A. R., Gould, N. I. M., and Toint, P. L., *Trust Region Methods*, SIAM, Philadelphia, PA, 2000.

[159] Kim, H., Ragon, S., Soremekun, G., Malone, B., and Sobieszczanski-Sobieski, J., "Flexible Approximation Model Approach for Bi-Level Integrated System Synthesis," *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, Aug. 2004, pp. 1–11, AIAA 2004-4545.

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

[161] Legresley, P. A. and Alonso, J. J., "Improving the Performance of Design Decomposition Methods with POD," *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, Aug. 2004, AIAA 2004-4465.

[162] Berkooz, G., Holmes, P., and Lumley, J. L., "The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows," *Annual Review of Fluid Mechanics*, Vol. 25, 1993, pp. 539–575. doi:10.1146/annurev.fl.25.010193.002543.

[163] Sobieszczanski-Sobieski, J., "Integrated System-of-Systems Synthesis," *AIAA Journal*, Vol. 46, No. 5, 2008, pp. 1072–1080. doi:10.2514/1.27953.

[164] Kim, H. M., *Target Cascading in Optimal System Design*, Ph.D. thesis, University of Michigan, 2001.

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

[166] Kim, H. M., Chen, W., and Wiecek, M. M., "Lagrangian Coordination for Enhancing the Convergence of Analytical Target Cascading," *AIAA Journal*, Vol. 44, No. 10, 2006, pp. 2197–2207. doi:10.2514/1.15326.

[167] Tosserams, S., Etman, L. F. P., Papalambros, P. Y., and Rooda, J. E., "An Augmented Lagrangian Relaxation for Analytical Target Cascading Using the Alternating Direction Method of Multipliers," *Structural and Multidisciplinary Optimization*, Vol. 31, No. 3, 2006, pp. 176–189. doi:10.1007/s00158-005-0579-0.

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

[169] Li, Y., Lu, Z., and Michalek, J. J., "Diagonal Quadratic Approximation for Parallelization of Analytical Target Cascading," *Journal of Mechanical Design*, Vol. 130, No. 5, 2008, pp. 051402—-1–11. doi:10.1115/1.2838334.

[170] Ruszczynski, A., "On Convergence of an Augmented Lagrangian Decomposition Method for Sparse Convex Optimization," *Mathematics of Operations Research*, Vol. 20, No. 3, 1995, pp. 634–656. doi:10.1287/moor.20.3.634.

[171] Han, J. and Papalambros, P. Y., "A Sequential Linear Programming Coordination Algorithm for Analytical Target Cascading," *Journal of Mechanical Design*, Vol. 132, No. 3, 2010, pp. 021003—-1–8. doi:10.1115/1.4000758.

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

[173] Chen, T.-Y., "Calculation of the Move Limits for the Sequential Linear Programming Method," *International Journal for Numerical Methods in Engineering*, Vol. 36, No. 15, 1993, pp. 2661–2679. doi:10.1002/nme.1620361510.

[174] Han, J. and Papalambros, P. Y., "A Note on the Convergence of Analytical Target Cascading with Infinite Norms," *Journal of Mechanical Design*, Vol. 132, No. 3, 2010, pp. 034502—-1–6. doi:10.1115/1.4001001.

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

[176] Kim, H. M., Kokkolaras, M., Louca, L. S., Delagrammatikas, G. J., Michelena, N. F., Filipi, Z. S., Papalambros, P. Y., Stein, J. L., and Assanis, D. N., "Target Cascading in Vehicle Redesign: A Class VI Truck Study," *International Journal of Vehicle Design*, Vol. 29, No. 3, 2002, pp. 199–225.

[177] Kim, H. M., Rideout, D. G., Papalambros, P. Y., and Stein, J. L., "Analytical Target Cascading in Automotive Vehicle Design," *Journal of Mechanical Design*, Vol. 125, No. 3, 2003, pp. 481–490. doi:10.1115/1.1586308.

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

[179] Tajima, J., Momiyama, F., and Yuhara, N., "A New Solution for Two-Bag Air Suspension System with Leaf Spring for Heavy-Duty Vehicle," *Vehicle System Dynamics*, Vol. 44, No. 2, 2006, pp. 107–138. doi:10.1080/00423110500385907.

[180] Chen, Y., Chen, X., and Lin, Y., "The Application of Analytical Target Cascading in Parallel Hybrid Electric Vehicle," *IEEE Vehicle Power and Propulsion Conference*, 2009, pp. 1602–1607.

[181] Han, J. and Papalambros, P. Y., "Optimal Design of Hybrid Electric Fuel Cell Vehicles Under Uncertainty and Enterprise Considerations," *Journal of Fuel Cell Science and Technology*, Vol. 7, No. 2, 2010, pp. 021020—-1–9. doi:10.1115/1.3179762.

[182] Allison, J. T., Walsh, D., Kokkolaras, M., Papalambros, P. Y., and Cartmell, M., "Analytical Target Cascading in Aircraft Design," *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, Jan. 2006, AIAA 2006-1325.

[183] Li, Z., Kokkolaras, M., Papalambros, P. Y., and Hu, S. J., "Product and Process Tolerance Allocation in Multistation Compliant Assembly Using Analytical Target Cascading," *Journal of Mechanical Design*, Vol. 130, No. 9, 2008, pp. 091701—-1–9. doi:10.1115/1.2943296.

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

[185] Michalek, J. J., Feinberg, F. M., and Papalambros, P. Y., "Linking Marketing and Engineering Product Design Decisions via Analytical Target Cascading," *Journal of Product Innovation Management*, Vol. 22, No. 1, 2005, pp. 42–62. doi:10.1111/j.0737-6782.2005.00102.x.

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

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

[188] Kokkolaras, M., Mourelatos, Z. P., and Papalambros, P. Y., "Design Optimization of Hierarchically Decomposed Multilevel Systems Under Uncertainty," *Journal of Mechanical Design*, Vol. 128, No. 2, 2006, pp. 503–508. doi:10.1115/1.2168470.

[189] Liu, H., Chen, W., Kokkolaras, M., Papalambros, P. Y., and Kim, H. M., "Probabilistic Analytical Target Cascading: A Moment Matching Formulation for Multilevel Optimization Under Uncertainty," *Journal of Mechanical Design*, Vol. 128, No. 4, 2006, pp. 991–1000. doi:10.1115/1.2205870.

[190] Tosserams, S., Etman, L. F. P., and Rooda, J. E., "Block-Separable Linking Constraints in Augmented Lagrangian Coordination in MDO," *Structural and Multidisciplinary Optimization*, Vol. 37, No. 5, 2009, pp. 521–527. doi:10.1007/s00158-008-0244-5.

[191] DeMiguel, A.-V., *Two Decomposition Algorithms for Nonconvex Optimization Problems with Global Variables*, Ph.D. thesis, Stanford University, 2001.

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

[193] Shin, M. K., Kang, B. S., and Park, G. J., "Application of the Multidisciplinary Design Optimization Algorithm to the Design of a Belt-Integrated Seat While Considering Crashworthiness," *Journal of Automobile Engineering*, Vol. 219, No. 11, 2005, pp. 1281–1292. doi:10.1243/095440705X34928.

[194] Liu, B., Haftka, R. T., and Watson, L. T., "Global-Local Structural Optimization Using Response Surfaces of Local Optimization Margins," *Structural and Multidisciplinary Optimization*, Vol. 27, No. 5, 2004, pp. 352–359. doi:10.1007/s00158-004-0393-0.

[195] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet," *Journal of Aircraft*, Vol. 41, No. 3, 2004, pp. 523–530. doi:10.2514/1.11478.

[196] Kennedy, G., Martins, J. R. R. A., and Hansen, J. S., "Aerostructural Optimization of Aircraft Structures Using Asymmetric Subspace Optimization," *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, BC, Canada, Sept. 2008, AIAA 2008-5847.

[197] Kaufman, M., Balabanov, V., Burgee, S. L., Giunta, A. A., Grossman, B., Haftka, R. T., Mason, W. H., and Watson, L. T., "Variable-Complexity Response Surface Approximations for Wing Structural Weight in HSCT Design," *Computational Mechanics*, Vol. 18, 1996, pp. 112–126. doi:10.1007/BF00350530.

[198] Hosder, S., Watson, L. T., Grossman, B., Mason, W. H., Kim, H., Haftka, R. T., and Cox, S. E., "Polynomial Response Surface Approximations for the Multidisciplinary Design Optimization of a High Speed Civil Transport," *Optimization and Engineering*, Vol. 2, 2001, pp. 431–452. doi:10.1023/A:1016094522761.

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

[200] Allison, J. T., Kokkolaras, M., and Papalambros, P. Y., "On Selecting Single-Level Formulations for Complex System Design Optimization," *Journal of Mechanical Design*, Vol. 129, 2007, pp. 898–906. doi:10.1115/1.2747632.

[201] Martins, J. R. R. A., Marriage, C., and Tedford, N., "pyMDO: An Object-Oriented Framework for Multidisciplinary Design Optimization," *ACM Transactions on Mathematical Software*, Vol. 36, No. 4, 2009, pp. 20:1–25. doi:10.1145/1555386.1555389.

[202] Gray, J., Moore, K. T., and Naylor, B. A., "OpenMDAO: An Open Source Framework for Multidisciplinary Analysis and Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, TX, Sept. 2010, AIAA 2010-9101.

[203] Lu, Z. and Martins, J. R. R. A., "Graph Partitioning-Based Coordination Methods for Large-Scale Multidisciplinary Design Optimization Problems," *Proceedings of the 14th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Indianapolis, IN, 2012.

[204] Padula, S. L. and Gillian, R. E., "Multidisciplinary Environments: A History of Engineering Framework Development," *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Porthsmouth, VA, Sept. 2006, AIAA 2006-7083.

[205] Kodiyalam, S. and Sobieszczanski-Sobieski, J., "Multidisciplinary Design Optimization - Some Formal Methods, Framework Requirements, and Application to Vehicle Design," *International Journal of Vehicle Design*, Vol. 25, 2001, pp. 3–22. doi:10.1504/IJVD.2001.001904.

[206] Kodiyalam, S., Kremenetsky, M., and Posey, S., "Balanced HPC Infrastructure for CFD and Associated Multi-Discipline Simulations of Engineering Systems," *Journal of Aerospace Sciences and Technologies*, Vol. 61, No. 3, 2009, pp. 434–443.

[207] Tosserams, S., Etman, L. F. P., and Rooda, J. E., "A Micro-Accelerometer MDO Benchmark Problem," *Structural and Multidisciplinary Optimization*, Vol. 41, No. 2, 2010, pp. 255–275. doi:10.1007/s00158-009-0422-0.

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

[209] Padula, S. L., Alexandrov, N., and Green, L. L., "MDO Test Suite at NASA Langley Research Center," *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, Sept. 1996, AIAA 1996-4028.