Vol. 40, No. 2, February 2002

Analytical and Computational Aspects of Collaborative Optimization for Multidisciplinary Design

Natalia M. Alexandrov*
NASA Langley Research Center, Hampton, Virginia 23681-2199
and
Robert Michael Lewis†
College of William and Mary, Williamsburg, Virginia 23187-8795

Analytical features of multidisciplinary optimization (MDO) problem formulations have significant practical consequences for the ability of nonlinear programming algorithms to solve the resulting computational optimization problems reliably and efficiently. We explore this important but frequently overlooked fact using the notion of disciplinary autonomy. Disciplinary autonomy is a desirable goal in formulating and solving MDO problems; however, the resulting system optimization problems are frequently difficult to solve. We illustrate the implications of MDO problem formulation for the tractability of the resulting design optimization problem by examining a representative class of MDO problem formulations known as collaborative optimization. We also discuss an alternative problem formulation, distributed analysis optimization, that yields a more tractable computational optimization problem.

Nomenclature

A_i	=	disciplinary analysis i
a_i	=	output of A_i
c_j	=	interdisciplinary consistency constraints
D_i	=	discipline i
f	=	system objective function
g_i	=	design constraints for D_i
l_i	=	design variables local to D_i
min	=	minimize
s	=	design variables shared by disciplines
s.t.	=	subject to

Introduction

THE analytical features of multidisciplinary optimization (MDO) problem formulations have significant consequences for the ability of nonlinear programming algorithms to solve the resulting computational optimization problems reliably and efficiently. This frequently overlooked fact is the theme of this and related papers.[1–4]

We illustrate the practical computational implications of problem formulation using collaborative optimization (CO).[5–8] CO is characterized by a distributed, bilevel structure, wherein a system problem seeks to optimize system performance, whereas disciplinary problems attempt to minimize the interdisciplinary inconsistency in the variables and responses shared by the disciplines.

The ideas underlying CO are intuitively appealing and are based on reasonable motivations. Historical evolution of engineering disciplines and the complexity MDO suggest that disciplinary autonomy is a desirable goal in formulating and solving MDO problems. Consequently, is not surprising that bilevel approaches that maintain a measure of disciplinary autonomy have appeared and reappeared in many forms over the past three decades. However, as we discuss, difficulties necessarily arise in solving the resulting computational optimization problems in theory and in practice. Difficulties in solving problems with CO and related methods have been observed by a number of researchers, including Thareja and Haftka,[9] Cormier et al.,[10] Giesing and Barthelemy,[11] and Kodiyalam.[12] We point out that they derive from the intrinsic mathematical properties of CO. Our line of inquiry is constructive because it clarifies practical computational issues in MDO. The discussion is intended for the engineering audience, although it contains some unavoidable mathematical details required to explain what are ultimately issues of a mathematical nature.

This line of analysis has immediate implications for a practitioner of MDO in that it describes and explains the obstacles one is likely to encounter in applying conventional optimization methods to distributed formulations. The obstacles can be overcome in one of two ways. One may wish to pursue the development of new optimization algorithms, specially suited to solving distributed optimization problems. In the absence of suitable algorithms, our analysis would indicate when and why one would instead pursue alternative problem formulations. We give one example of an alternative class of formulations that possesses many of the attractive features of distributed optimization approaches, while avoiding their computational difficulties. We conjecture that an ideal MDO problem formulation—one that manifests complete disciplinary autonomy and leads to optimization problems that can be solved efficiently and reliably—may not be possible. This observation emphasizes the need for recognizing the tradeoffs among various features of problem formulations and their computational consequences.

The analysis presented here supports a wider program we call the *algorithmic perspective* on MDO problem synthesis. It takes as its starting point the abilities—and inabilities—of optimization algorithms and seeks to formulate MDO problems so that the resulting optimization problems can be solved reliably and efficiently, reflecting the organizational and physical features of the application to maximum extent without sacrificing solubility by available algorithms. The study of the analytical and computational aspects of MDO problem formulations is central to this program.

The algorithmic perspective stands in contrast to the conventional approach to MDO we call the structural perspective, wherein the primary considerations are the physical or organizational characteristics of the system being designed. Concerns about the resulting optimization problem are secondary. Although successful for many specific problems, the structural approach frequently leads

Received 19 May 2000; revision received 2 July 2001; accepted for publication 3 July 2001.

*Research Scientist, Multidisciplinary Optimization Branch, Aerospace Systems Concepts and Analysis, Mail Stop 159; n.alexandrov@larc.nasa.gov. Member AIAA.

†Assistant Professor, Department of Mathematics, P.O. Box 8795; buckaroo@math.wm.edu. Member AIAA.

to what several engineering researchers have described to us as one-of-a-kind problem solutions. That is, the choice of formulation can give rise to computational problems that require great effort in implementing and tuning the formulation and algorithms until a satisfactory solution is attained. Even if the underlying ideas suffer from serious deficiencies, highly customized approaches can usually be made to yield results, with sufficient effort. However, this means, in effect, developing a new, custom approach for each application. Because successes of a particular method are reported, whereas failures are usually not, the literature reflects a level of systematic success and effectiveness that can be misleading.

In this paper we support our observations with analytical results, illustrated by simple examples. The analysis is by no means exhaustive; we address only some of the analytical and computational features of immediate practical import. Furthermore, a number of interesting MDO problem formulations are not discussed here because we do not intend this paper as a survey. Instead, our intent is to bring to light the practical impact of the analytical features of MDO problem formulation on computational tractability of the resulting design optimization problem. CO has been chosen because it provides a particularly illuminating example.

Model Problem

For ease of exposition, we present our discussion for a two-discipline model problem. The disciplines might represent the aeroelastic interaction between aerodynamics D_1 and structural analysis D_2 for a wing in steady-state flow. The discussion is applicable to MDO problems with an arbitrary number of disciplines, however.

Each disciplinary subsystem D_i is based on a disciplinary analysis A_i that takes as its input a set of design variables (s, l_i) and parameters derived from some or all of the outputs from the other disciplinary analysis. The system design variables s are shared by both disciplines. The disciplinary design variables l_1 and l_2 are local to D_1 and D_2, respectively. The outputs a_i of each analysis include all data passed to the other discipline as parameters as well as quantities passed to design constraints and objectives. In our aeroelastic example the information a_2, passed from structures to aerodynamics, would include the wing shape. The information a_1, passed from aerodynamics to structures, would include the aerodynamic loads. The parameters derived from the analysis outputs a_j, $j \neq i$ of the other discipline are not directly manipulated by the designer in D_i.

The coupled multidisciplinary analysis system (MDA) reflects the physical requirement that a solution simultaneously satisfy A_1 and A_2. Given (s, l_1, l_2), we write the MDA as the system

$$a_1 = A_1(s, l_1, a_2) \tag{1}$$

$$a_2 = A_2(s, l_2, a_1) \tag{2}$$

A_1 and A_2 are independently soluble: given (s, l_i, a_j), we can compute the output a_i via Eq. (1) or (2). The MDA thus implicitly defines a_1 and a_2 as functions of (s, l_1, l_2): $a_1 = a_1(s, l_1, l_2)$ and $a_2 = a_2(s, l_1, l_2)$.

The disciplinary design constraints $g_1(s, l_1, a_1)$ and $g_2(s, l_2, a_2)$ explicitly depend only on a single discipline's output. No constraint involves a_1 and a_2 jointly. This assumption simplifies the exposition, but is not essential.

A conventional approach to MDO problem formulation, commonly used in engineering design, is to impose an optimizer on the MDA. Given the need to satisfy the MDA at a solution, this approach is natural. We call it the fully integrated optimization (FIO) formulation and use it to give a canonical statement of the problem one wishes to solve. FIO is depicted in Fig. 1. Its mathematical statement is

$$\begin{aligned} \min_{s,l_1,l_2} \quad & f[s, l_1, l_2, a_1(s, l_1, l_2,), a_2(s, l_1, l_2)] \\ \text{s.t.} \quad & g_1[s, l_1, a_1(s, l_1, l_2)] \geq 0 \\ & g_2[s, l_2, a_2(s, l_1, l_2)] \geq 0 \end{aligned} \tag{3}$$

where the evaluation of the objective and constraints requires solving MDA (1) and (2) for the disciplinary analysis outputs $a_1(s, l_1, l_2)$ and $a_2(s, l_1, l_2)$.

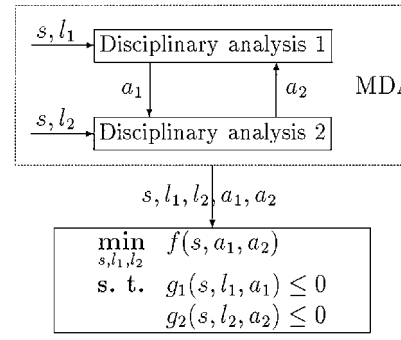

Fig. 1 Fully integrated optimization.

Distributed Formulations

One of the main issues in MDO is problem synthesis, i.e., integration of the component disciplines into a tractable optimization problem. One wishes to maintain the inherent disciplinary autonomy because organizational and computational considerations make it desirable to keep the transfer of information among the disciplines to a minimum. In practice, disciplinary autonomy can be attained in the following ways:

1) Some distributed methods attempt to avoid the necessity of developing a separate MDA capability. The feasibility of all optimization iterates with respect to MDA (1) and (2) is the main benefit of FIO: even if optimization cannot continue to a solution because of limited resources, the intermediate designs will be physically realizable. However, an MDA capability is not usually developed at the same time as the constituent disciplinary analyses. Developing an MDA capability together with the ability to compute design sensitivities that depend on the MDA presents a laborious and time-consuming task. This drawback of FIO strongly motivates the development of distributed formulations under study here.

2) FIO leads to the presence of the complete set of local disciplinary design variables l_i in the system problem (3). Some distributed formulations use subsystem optimization problems to eliminate the local design variables from the system problem, thus hiding the details of the subsystem design from the system problem.

Discrepancy functions[13,14] suggest themselves as a device for enabling disciplinary autonomy by distributing a coupled problem as autonomous subproblems. The idea is as follows. MDO problems are inherently coupled, with feedback among the disciplines in the form of shared variables and responses. Solving a problem means that a single set of shared variables and responses must satisfy the disciplinary analyses and disciplinary design constraints simultaneously at solutions. Partitioning a problem into a set of subproblems usually means that at intermediate solutions an inconsistency exists between some or all of the shared variables and responses. This inconsistency can be measured in many different ways as a scalar function, with the actual form depending on the quantity in question. We call such scalar measures of inconsistency *discrepancy functions* (DF). Formulations based on DF attempt to remove the interdisciplinary inconsistency at solutions, usually by requiring that the DF be zero there. The resulting formulations lead to bilevel or multilevel optimization problems.

Decomposition and information flow in such bilevel approaches are intended to mirror those present in engineering organizations. A system coordination problem attempts to optimize the system objective. In the process of doing so, it issues design targets to the component disciplines. In the subsystem problems the disciplines must design to match these targets, i.e., try to reduce that discipline's DF to zero. This is one sense in which distributed bilevel formulations can be viewed as respecting disciplinary autonomy.

DF-based formulations are also motivated by the wish to keep the designs feasible with respect to the corresponding disciplinary design constraints during optimization. This avoids problems with designs that cause a breakdown of disciplinary analyses. However, because the overall design does not, in general, satisfy the system interdisciplinary consistency constraints, stopping in the middle of optimization can yield a design that is not physically consistent.

Approaches based on DF include optimization by linear decomposition (OLD) or hierarchical decomposition.[13–19] CO is a related approach; the underlying idea has a long history.[13,14,19–24] In both approaches local design variables are eliminated from the system problem and manipulated directly only in disciplinary problems. Furthermore, CO dispenses with an explicit MDA. In OLD the disciplines are given the autonomous task of minimizing disciplinary design infeasibility while maintaining system consistency. The system problem is to drive the design infeasibility to zero. In CO the situation is reversed: the disciplines are given the autonomous task of minimizing system inconsistency while maintaining disciplinary design feasibility. In both approaches optimization of the system objective, subject to interdisciplinary consistency, is performed in the system problem.

Although bilevel approaches based on DF are intuitively appealing and reasonable, the resulting optimization problems are intrinsically difficult to solve by conventional optimization methods, as we discuss presently. The observations on the differences in computational behavior of two formulations of the same problem illustrate the importance of considering practical algorithmic consequences of choosing a specific MDO problem formulation.

Collaborative Optimization

To formulate problem (3) along the lines of CO, we introduce new disciplinary design variables σ_1, σ_2. In the subsystem problems they serve as local copies of the shared variables s, thus relaxing the coupling through s. CO is a bilevel optimization approach in which the system coordination problem attempts to optimize the system objective subject to N interdisciplinary consistency constraints $C = \{c_1, \ldots, c_N\}$:

$$\min_{s,t_1,t_2} \quad f(s, t_1, t_2), \qquad \text{s.t.} \quad C(s, t_1, t_2) = 0 \tag{4}$$

Problem (4) controls the system design variables s and interdisciplinary coupling variables (t_1, t_2), which are issued as design targets for the disciplinary inputs and outputs a_1 and a_2 to the constituent disciplines. In the subsystem problems the disciplines must design to match these targets as follows. In D_1, given (s, t_1, t_2), we compute $\bar{\sigma}_1(s, t_1, t_2)$ and $\bar{l}_1(s, t_1, t_2)$ as solutions of the following minimization problem in (σ_1, l_1):

$$\begin{aligned} \min_{\sigma_1,l_1} \quad & \tfrac{1}{2}\big[\|\sigma_1 - s\|^2 + \|a_1(\sigma_1, l_1, t_2) - t_1\|^2\big] \\ \text{s.t.} \quad & g_1[\sigma_1, l_1, a_1(\sigma_1, l_1, t_2)] \geq 0 \end{aligned} \tag{5}$$

where a_1 is computed via the disciplinary analysis $a_1 = A_1(\sigma_1, l_1, t_2)$. An analogous problem for D_2 defines solutions $\bar{\sigma}_2(s, t_1, t_2)$ and $\bar{l}_2(s, t_1, t_2)$ of the problem

$$\begin{aligned} \min_{\sigma_2,l_2} \quad & \tfrac{1}{2}\big[\|\sigma_2 - s\|^2 + \|a_2(\sigma_2, l_2, t_1) - t_2\|^2\big] \\ \text{s.t.} \quad & g_2[\sigma_2, l_2, a_2(\sigma_2, l_2, t_1)] \geq 0 \end{aligned} \tag{6}$$

with a_2 computed via $a_2 = A_2(\sigma_2, l_2, t_1)$.

The objectives in the subsystem problems are discrepancy functions. The introduction of subsystem problems of the forms (5) and (6) is a distinctive characteristic of CO. The problems can be solved autonomously. In solving them, we eliminate l_i from the system problem and decouple the calculation of the disciplinary analysis outputs a_i. Information from the solutions of problems (5) and (6) is then used to define the system consistency constraints c_i.

In one instance of CO,[5,6,8,20] the consistency condition drives to zero the minimum value of the DF in problems (5) and (6). At the system level the interdisciplinary consistency constraints $C = (c_1, c_2)$ are simply the optimal values of the objectives in problems (5) and (6):

$$\begin{aligned} c_1(s, t_1, t_2) &= \tfrac{1}{2}\Big\{\|\bar{\sigma}_1(s, t_1, t_2) - s\|^2 \\ &\quad + \|a_1[\bar{\sigma}_1(s, t_1, t_2), \bar{l}_1(s, t_1, t_2), t_2] - t_1\|^2\Big\} \\ c_2(s, t_1, t_2) &= \tfrac{1}{2}\Big\{\|\bar{\sigma}_2(s, t_1, t_2) - s\|^2 \\ &\quad + \|a_2[\bar{\sigma}_2(s, t_1, t_2), \bar{l}_2(s, t_1, t_2)t_1] - t_2\|^2\Big\} \end{aligned} \tag{7}$$

where the bars over $\bar{\sigma}_1, \bar{\sigma}_2, \bar{l}_1, \bar{l}_2$ indicate that these values are the results of solving the subsystem problems for the given value of the system variables. We call this version CO_2, where 2 refers to the fact that the c_i are sums of squares.

An alternative consistency condition gives rise to a second instance of CO (denoted CO_1), where the system variables are matched directly with their subsystem counterparts computed in problems (5) and (6). The consistency constraints $C = (c_1, \ldots, c_4)$ are

$$\begin{aligned} c_1(s, t_1, t_2) &= \bar{\sigma}_1(s, t_1, t_2) - s \\ c_2(s, t_1, t_2) &= a_1[\bar{\sigma}_1(s, t_1, t_2), \bar{l}_1(s, t_1, t_2), t_2] - t_1 \\ c_3(s, t_1, t_2) &= \bar{\sigma}_2(s, t_1, t_2) - s \\ c_4(s, t_1, t_2) &= a_2[\bar{\sigma}_2(s, t_1, t_2), \bar{l}_2(s, t_1, t_2), t_1] - t_2 \end{aligned} \tag{8}$$

Note that (c_1, c_2) are associated with D_1, and (c_3, c_4) are associated with D_2.

In either approach a value of the system variables (s, t_1, t_2) is realizable for D_i if the optimal value of the DF in the corresponding disciplinary subsystem problem (5) or (6) is zero. Realizable values of s, t_1, t_2 correspond to desirable designs: D_i can exactly match the system targets without violating the disciplinary design constraints. In general, there are many realizable values of the system variables for a given discipline. A point (s, t_1, t_2) is feasible for the system problem when it is realizable for all of the constituent disciplines.

Examples of Reformulation

Two examples illustrate the analysis of CO.[2] The simplicity is a conscious choice. When complex problems are used as the only test of methodology, it is difficult to distinguish the behavior caused by the intrinsic properties of the method from those features caused by the various aspects of the problem and implementation. Simple examples allow us to isolate the intrinsic properties of problem formulations. Moreover, simple problems provide a lower bound on the reliability of a solution technique: although it is clear that MDO methods are not intended for very small and simple problems, any practical optimization approach should be able to solve such problems reliably. Experience with CO applied to more complex problems is reported elsewhere.[9,10,12,25]

Examples (9) and (10) are trivially solved by conventional optimization methods. This feature removes the issue of the problem implementation complexity and isolates the intrinsic properties of problem formulations under consideration here, as well as their algorithmic consequences.

Our first example is exceedingly simple:

$$\min_{s} \quad f(s) = s, \qquad \text{s.t.} \quad 0 \leq s \leq 1 \tag{9}$$

Our second example has a convex quadratic objective and linear constraints:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\big[a_1^2(l_1, l_2) + 10\,a_2^2(l_1, l_2)\big] \\ \text{s.t.} \quad & s + l_1 \leq 1 \\ & -s + l_2 \leq -2 \end{aligned} \tag{10}$$

where (a_1, a_2) solves $2a_1 + a_2 = l_1$ and $a_1 + 2a_2 = l_2$.

To reformulate problem (9) along the lines of CO, we create two "disciplines" associated with the constraints $s \geq 0$ and $s \leq 1$, viewed as two disciplinary design constraints. Given s, the subsystem problems are

$$\begin{aligned} &\min_{\sigma_1} \quad \tfrac{1}{2}\|\sigma_1 - s\|^2, \qquad \min_{\sigma_2} \quad \tfrac{1}{2}\|\sigma_2 - s\|^2 \\ &\text{s.t.} \quad \sigma_1 \geq 0 \qquad\qquad \text{s.t.} \quad \sigma_2 \leq 1 \end{aligned} \tag{11}$$

The solutions, as functions of s, are

$$\bar{\sigma}_1(s) = \begin{cases} 0 & \text{if} \quad s \leq 0, \\ s & \text{if} \quad s \geq 0 \end{cases} \qquad \bar{\sigma}_2(s) = \begin{cases} s & \text{if} \quad s \leq 1 \\ 1 & \text{if} \quad s \geq 1 \end{cases} \tag{12}$$

The CO_2 system problem is then

$$\begin{aligned} \min_{s} \quad & s \\ \text{s.t.} \quad & c_1(s) = \tfrac{1}{2}\|\bar{\sigma}_1(s) - s\|^2 = 0 \\ & c_2(s) = \tfrac{1}{2}\|\bar{\sigma}_2(s) - s\|^2 = 0 \end{aligned} \tag{13}$$

whereas the CO_1 system problem is

$$\begin{aligned} \min_{s} \quad & s \\ \text{s.t.} \quad & c_1(s) = \bar{\sigma}_1(s) - s = 0 \\ & c_2(s) = \bar{\sigma}_2(s) - s = 0 \end{aligned} \tag{14}$$

Reformulating problem (10) along the lines of CO, we obtain the system-level problem

$$\min_{s,t_1,t_2} \tfrac{1}{2}\left(t_1^2 + 10\,t_2^2\right), \qquad \text{s.t.} \quad C(s, t_1, t_2) = 0$$

which becomes CO_2 if $C = (c_1, c_2)$ is defined as in Eqs. (7) and CO_1 if $C = (c_1, \ldots, c_4)$ is defined as in Eqs. (8). Given (s, t_1, t_2), the constrained optimal values $\bar{\sigma}_1(s, t_1, t_2)$ and $\bar{l}_1(s, t_1, t_2)$ are computed by solving problem (5) for D_1 with the disciplinary constraint $\sigma_1 + l_1 \le 1$, where a_1 solves the disciplinary analysis $2a_1 + t_2 = l_1$. Similarly, for D_2 we compute $\bar{\sigma}_2(s, t_1, t_2)$ and $\bar{l}_2(s, t_1, t_2)$ via problem (6) with the constraint $-\sigma_2 + l_2 \le -2$, where a_2 solves the disciplinary analysis $t_1 + 2a_2 = l_2$. The subsystem problem solutions are

$$\begin{aligned} \bar{\sigma}_1(s, t_1, t_2) &= s + \tfrac{1}{5}\min[(-s - 2t_1 - t_2 + 1), 0] \\ \bar{l}_1(s, t_1, t_2) &= 2t_1 + t_2 + \tfrac{4}{5}\min[(-s - 2t_1 - t_2 + 1), 0] \\ \bar{\sigma}_2(s, t_1, t_2) &= s + \tfrac{1}{5}\max[(-s + t_1 + 2t_2 + 2), 0] \\ \bar{l}_2(s, t_1, t_2) &= t_1 + 2t_2 - \tfrac{4}{5}\max[(-s + t_1 + 2t_2 + 2), 0] \end{aligned} \tag{15}$$

Analytical Features of CO

In this section we discuss and illustrate some of the more pronounced analytical features of CO and their computational consequences. A more detailed discussion can be found elsewhere.[2] We believe that the analysis explains many of the reported computational difficulties.[9,12,25] The features we discuss make it harder for conventional algorithms to solve the CO system problem and also degrade the efficiency with which the problems will be solved. Moreover, bad things happen at good points: the difficulties necessarily arise at values of the system variables that are realizable for individual disciplines and, more specifically, designs and disciplinary inputs and outputs that correspond to a consistent MDA. This is an unavoidable consequence of the way CO eliminates l_i from the system problem.

Breakdown of the Stationarity Conditions in CO_2

The system problem in CO_2 fails to satisfy the standard Karush–Kuhn–Tucker (KKT) stationarity conditions for a constrained minimizer because, in general, Lagrange multipliers do not exist for the system problem. Example (9) illustrates this property. The gradients of the system consistency constraints are

$$\nabla c_1(s) = \begin{cases} s & \text{if} \quad s \le 0, \\ 0 & \text{if} \quad s \ge 0 \end{cases} \qquad \nabla c_2(s) = \begin{cases} 0 & \text{if} \quad s \le 1 \\ s & \text{if} \quad s \ge 1 \end{cases}$$

The minimizer of the system problem (13) is $s_* = 0$, and $\nabla c_1(s_*) = \nabla c_2(s_*) = 0$. The KKT conditions for problem (13) would require the existence of Lagrange multipliers λ_1^*, λ_2^* such that

$$\nabla f(s_*) + \lambda_1^* \nabla c_1(s_*) + \lambda_2^* \nabla c_2(s_*) = 0$$

However, we have

$$\nabla f(s_*) + \lambda_1^* \nabla c_1(s_*) + \lambda_2^* \nabla c_2(s_*) = \nabla f(s_*) = 1$$

In general, the KKT necessary condition for a point x_* to be a (local) minimizer of a generic equality constrained optimization problem $\{\min_x f(x) | C(x) = 0\}$ is that $C(x_*) = 0$ and there exists a vector of Lagrange multipliers λ_* for which $\nabla f(x_*) + \nabla C(x_*)\lambda_* = 0$. If x_* is feasible and $\nabla C(x_*) = 0$, then the KKT stationarity condition holds if and only if $\nabla f(x_*) = 0$. Thus, if the constraint Jacobian vanishes at x_* the Lagrange multiplier rule will not hold at x_*, unless x_* is also an unconstrained stationary point: $\nabla f(x_*) = 0$. In general, this is not the case.

Unfortunately, this situation necessarily arises in the system problem of CO_2. The system constraints are differentiable at system feasible points; however, at values of the system variables that are realizable for a given discipline the gradient of the CO_2 system constraints associated with that discipline vanishes. That is, if $c_i(s, t_1, t_2) = 0$ then $\nabla c_i(s, t_1, t_2) = 0$. This means that the Jacobian of the system constraints will drop rank whenever the system variables become realizable for one or more of the disciplines, which can cause numerical algorithms to fail at realizable values of the system variables. Another consequence is that the Jacobian of the system equality constraints in CO_2 vanishes at every feasible point of the system problem. This, in turn, implies that Lagrange multipliers do not exist, in general, for the system problem in CO_2.

The nonexistence of Lagrange multipliers manifests itself in a number of practical difficulties. For instance, solutions to the system problem exist, but we cannot identify them, and this inability to characterize solutions numerically via the KKT conditions has unfortunate consequences for computation. Assumptions about the validity of the KKT conditions underlie the ways in which optimization algorithms compute steps, gauge progress, and make decisions about termination, among other things. One practical feature is that one could begin an optimization algorithm at or near a solution, but, because the KKT conditions do not hold, the algorithm will move away from the solution, leaving the feasible region, and return to it only later.

A related feature is that algorithms that use augmented Lagrangians or similar merit functions to decide whether to accept an iterate can break down because, e.g., the penalty weights in the problem merit function grow without bound.

Moreover, because the KKT conditions do not hold at solutions of the system optimization problem, we have no way to gauge the progress of a conventional optimization algorithm applied to the system problem. Once the optimization algorithm terminates, we cannot, say, look at the gradient of the Lagrangian to determine whether we are close to a minimizer.

The breakdown of the KKT conditions in the system problem is an inherent feature of CO_2. The difficulty is not caused by the intrinsic geometry of the system or disciplinary feasible regions. Rather, it lies in the representation of the feasible region in terms of system constraints in CO_2. The vanishing of the Jacobian has been observed previously,[5] but its consequences appear not to have been fully appreciated.

Simple tests illustrate how the breakdown of the KKT conditions in CO_2 can impede and even thwart computational optimization. Table 1 presents the behavior of a Sequential Quadratic Programming (SQP) algorithm, the NPSOL[26] package. (The use of

Table 1 Iteration history of NPSOL applied to the CO_2 system problem for problem (9) with $s_0 = 0.001$ (columns 2, 3) and $s_0 = -0.001$ (columns 4, 5)

Iteration	s	Penalty	s	Penalty
0	1.000e−03	0.0e+00	−1.000e−03	0.0e+00
1	−9.990e−01	4.2e+00	−1.000e−00	1.0e+00
2	−9.847e−01	5.7e+00	−9.857e−01	1.4e+00
3	−8.282e−01	7.4e+00	−8.290e−01	1.9e+00
4	−4.142e−01	2.7e+01	−4.145e−01	6.9e+00
5	−3.430e−01	5.9e+01	−3.432e−01	1.5e+01
6	−1.718e−01	4.0e+02	−1.716e−01	1.0e+02
7	−1.436e−01	8.2e+02	−1.434e−01	2.1e+02
8	−7.251e−02	5.4e+03	−7.170e−02	1.4e+03
9	−6.076e−02	1.1e+04	−5.992e−02	2.8e+03
10	−3.203e−02	6.5e+04	−2.996e−02	1.9e+04
11	−2.717e−02	1.2e+05	−2.503e−02	3.9e+04
12	−1.727e−02	5.1e+05	−1.252e−02	2.6e+05
13	−1.442e−02	1.9e+06	−1.046e−02	5.3e+05
14	−1.414e−02	4.7e+06	−5.230e−03	3.5e+06

names of commercial software in this paper is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products by NASA or ICASE.)

The solution is $s_* = 0$. The initial guesses are $s_0 = 0.001$, which is close to s_* and also feasible with respect to the system constraints, and $s_0 = -0.001$, which is also near s_* but slightly infeasible.

When we start at $s_0 = 0.001$, because the system constraints vanish in the interior of the feasible region $0 \le s \le 1$, the problem appears to be unconstrained at s_0, and so we immediately take a large step that produces s_1 violating the design constraints. We then spend the remainder of the iterations working our way back towards feasibility.

Note the column labeled "Penalty." It holds the penalty weights in the augmented Lagrangian used by NPSOL as a merit function to gauge progress. The large values of the penalty weight reflect the nonexistence of Lagrange multipliers for the system problem; the algorithm is compensating for the system constraint Jacobian vanishing by increasing the penalty parameter (in principle, without bound as it approaches the solution).

The behavior of NPSOL is even more striking for the CO_2 formulation of problem (10). The solution to the strictly convex original problem is unique. From some starting points [e.g., $(s, t_1, t_2) = (1, 1, 1)$] NPSOL applied to the CO_2 system problem finds the optimal solution:

$$s = 1.63\overline{63}, \qquad t_1 = -0.30\overline{30}, \qquad t_2 = -0.03\overline{03}$$

$$\bar{\sigma}_1(s, t_1, t_2) = \bar{\sigma}_2(s, t_1, t_2) = s$$

$$\bar{l}_1(s, t_1, t_2) = -0.63\overline{63}, \qquad \bar{l}_2(s, t_1, t_2) = -0.36\overline{36}$$

$$a_1(s, t_1, t_2) = t_1, \qquad a_2(s, t_1, t_2) = t_2 \tag{16}$$

with the associated optimal value function of 5.05×10^{-2}, although at a considerably greater computational cost (several hundred disciplinary analyses) than that of the solution of FIO. (In the CO_2 tests described, we compute system sensitivities via postoptimality sensitivity analysis of solutions of the disciplinary problems, as one might in practical application of CO_2, while the sensitivities used inside the disciplinary subproblems are computed analytically.) None of the iterates generated in the system problem are realizable for either discipline. This nonrealizability is actually the favorable situation from the perspective of applying a numerical optimization algorithm to the CO_2 system problem. However, intermediate designs do not satisfy the physical constraints.

The worst behavior is observed when starting from values of the system variables that are feasible with respect to the system constraints (which is at odds with what one would hope for). For instance, if we start at $(s, t_1, t_2) = (-3, -3, -3)$ NPSOL terminates after a few system iterations (and a cost of over 200 analyses for each discipline) at

$$s = -2.806, \qquad t_1 = -5.658, \qquad t_2 = 0.301 \tag{17}$$

with the associated objective value of 16.46. The final system variables (and all intermediate iterates) satisfy the system consistency constraints, which is the adverse situation in CO because the computed Jacobians at this point are singular, and this causes NPSOL to fail. The singularity is reflected in the final estimate of -1.67×10^{10} for the Lagrange multiplier associated with the system constraint c_1.

Starting much closer to the solution, but still feasible with respect to the system constraints, we may fail to converge to the solution, e.g., when started from $(s, t_1, t_2) = (1.63, -0.3333, -0.0333)$, NPSOL halts, unable to make further progress, at a feasible value of the system variables with an objective of 6.10×10^{-2}, over 20% greater than the optimal value and only a slight improvement on an initial objective value of 6.11×10^{-2}.

As already noted, because the KKT conditions do not hold at solutions of the system problem we cannot reliably use conventional metrics to determine whether putative answers are nearly stationary and thus close to a solution. For instance, if we start with the system values $(s, t_1, t_2) = (1.63, -0.302, -0.302)$ we terminate at a point that appears to NPSOL to be a KKT point: It is feasible, and the projection of the objective gradient onto the linearization of the active constraints is small in magnitude. However, the Jacobian is nearly zero, and so small errors in computing the system sensitivities make the Jacobian mostly noise and the projection is meaningless.

The examples illustrate a transformation of simple, smooth, convex optimization problems with small numbers of variables into problems that are difficult to solve. Moreover, the analytical reasons for the computational difficulties are inherent in the transformation, and so they will not disappear for problems of greater size or complexity.

Breakdown of the Stationarity Conditions in CO_1

CO_1 is motivated by the need to alleviate the performance difficulties of CO_2,[5] but its use presents its own difficulties. In particular, the derivatives of the CO_1 system constraints associated with a given discipline are necessarily discontinuous at the boundary of the feasible region for that discipline.

For instance, the system constraints for (9) are

$$c_1(s) = \begin{cases} -s & \text{if} \quad s \le 0, \\ 0 & \text{if} \quad s \ge 0 \end{cases} \qquad c_2(s) = \begin{cases} 0 & \text{if} \quad s \le 1 \\ 1 - s & \text{if} \quad s \ge 1 \end{cases}$$

both of which have discontinuous derivatives: the first at $s = 0$, and the second at $s = 1$. These points correspond to the boundaries of the disciplinary feasible regions $\{s \mid s \ge 0\}$ and $\{s \mid s \le 1\}$.

A similar lack of differentiability can be seen in the reformulation of problem (10). The solutions of the disciplinary subproblems are given in Eqs. (15); the presence of the min and max terms makes these solutions nondifferentiable at values of the system variables along the boundary of the realizable sets for each discipline.

The discontinuity of the system derivatives is a general feature of CO_1 and is not peculiar to examples (9) and (10). Each subsystem problem (5) and (6) minimizes the distance from the disciplinary feasible region to the target values of the system design and coupling variables. For target values of the system variables corresponding to designs at the boundary of a disciplinary feasible region, the solution of the corresponding disciplinary optimization problem undergoes an abrupt and nondifferentiable change. Moreover, this discontinuity of the derivatives will, in general, occur at the solution to the system problem because at least one disciplinary design constraint will be binding at the solution, in general. That is, we can expect the solution of the system problem to be on the boundary of one (or more) of the feasible regions for the individual disciplines, and at such points the CO_1 constraints have discontinuous derivatives.

Example (10) demonstrates this effect. NPSOL finds the solution in four to five iterations, regardless of the starting point. NPSOL applied to the CO_1 reformulation, on the other hand, behaves erratically. If we start from the initial point $(s, t_1, t_2) = (1.63, -0.302, -0.0302)$, which is close to the exact solution, $(1.63, -0.30, -0.03)$, NPSOL finds the solution at a cost of about 50 disciplinary optimization problems for each discipline. If we start at the point $(s, t_1, t_2) = (-1, -1, -1)$, NPSOL terminates, unable to make further progress, at

$$s = -0.9969, \qquad t_1 = -1.4640, \qquad t_2 = -0.0689$$

$$\bar{\sigma}_1(s, t_1, t_2) = \bar{\sigma}_2(s, t_1, t_2) = s$$

$$\bar{l}_1(s, t_1, t_2) = -1.6018, \qquad \bar{l}_2(s, t_1, t_2) = -2.9969$$

$$a_1(s, t_1, t_2) = t_1, \qquad a_2(s, t_1, t_2) = t_2$$

This design satisfies the system constraints but has an associated objective value of 1.096. (The optimal objective value is 5.05×10^{-2}.) The associated disciplinary design variables for D_2 also lie on the boundary of the disciplinary feasible region (because $s + \bar{l}_2 = -2$), and so we encounter the discontinuity in the constraint Jacobian. Examination of the finite difference estimate of the Jacobian computed by NPSOL at this point reveals that the Jacobian is highly inaccurate.

Starting from $(s, t_1, t_2) = (0, 0, 0)$, NPSOL approaches but does not succeed in finding the correct answer of the original problem. We terminated this run after 500 system iterations with an objective

that was 2.7 times greater than the optimal value, at a cost of solving over 3000 disciplinary optimization problems for each discipline.

Additional Nonsmoothness

When the set of disciplinary design constraints binding at the solution of the subsystem problems (5) and (6) changes as a function of the system targets (s, t_1, t_2), there may be a discontinuity in the derivative of the system constraints for both CO_1 and CO_2. This difficulty is a well-known phenomenon that arises in the dependence of solutions of optimization problems on parameters and has been noted in connection with bilevel approaches to MDO.[5,16,27,28] The discontinuity of the constraint Jacobian in CO_1 is one manifestation of this phenomenon, in fact.

There is also potential for multiple local solutions of the subsystem problems (5) and (6). The computed solutions may fail to depend continuously on the system targets (s, t_1, t_2), and it is not clear how to ensure that one computes only a continuous branch of local minimizers to the subsystem problems.

We do not illustrate these two difficulties with examples. They are widely known and may or may not occur, depending on the particular problem and the solution algorithms. The characteristics described in the preceding sections, on the other hand, are intrinsic to CO and may occur even if the original problem is perfectly well-behaved.

Overdetermined System-Level Constraints

In a typical equality constrained minimization problem there are fewer equality constraints than optimization variables, i.e., the degrees of freedom in searching for a problem solution outnumber the binding constraints. Both CO_1 and CO_2 can lead to system problems that have more equality constraints than optimization variables, with CO_1 typically leading to more constraints than does CO_2. Example (9) illustrates this: the system problem has a single variable but two equality constraints.

Although the system equality constraints are consistent insofar as they are satisfied at any multidisciplinary consistent design (s, t_1, t_2), away from the solution the overdetermined constraints can cause trouble for standard optimization algorithms. For instance, in SQP the system constraints are linearized. Because there may be many more constraints than unknowns, the resulting linear system can appear to be overdetermined and without solution, leading to an infeasible SQP subproblem. This, in turn, can lead the optimization algorithm to wrongly conclude that the system problem is infeasible and to terminate without having found a solution. We have observed this behavior in practice.

In CO_2 one can try to avoid this problem by summing the constraints from different disciplines into a single nonlinear equality constraint. In CO_1 this remedy does not apply.

Increased Nonlinearity of the Transformed Problem

CO transforms originally smooth problems into nonsmooth ones with a higher degree of nonlinearity. For example, the original problem (9) is linear, whereas the resulting CO system and subsystem problems are nonlinear. In CO_2 the system problem involves piecewise quadratic constraints, whereas in CO_1 the system constraints are not continuously differentiable. The increased nonlinearity arises from the elimination of the local, disciplinary design variables via the disciplinary optimization subproblems.

As a general rule in nonlinear programming, it is important not to increase nonlinearity or introduce other structural complications when transforming problems.[29,30] For instance, practical optimization algorithms are often based on successive linearizations. Linearization is exact for linear problems, which allows algorithms to take advantage of this special structure. The CO reformulation does not preserve linearity in the system constraints. The practical difficulty of solving increasingly nonlinear system problems in CO is noted in computational experiments.[10]

Addressing Computational Difficulties

As we have seen, the nature of CO can make it difficult for numerical optimization algorithms to arrive at realizable, interdisciplinary consistent designs. We can attempt to address the difficulties via one of several relaxations. All relaxations, in effect, are accomplished by staying away from realizable designs.

Table 2 Results of relaxing the CO_2 constraints for problem (10), starting at $(-3, -3, -3)$ (columns 2, 3) and $(1, 1, 1)$ (columns 4, 5)[a]

ε	f_*	KKT?	f_*	KKT?
10^{-1}	4.75	No	1.62×10^{-20}	Yes
10^{-2}	4.79	No	6.82×10^{-3}	Yes
10^{-3}	4.79	No	3.23×10^{-2}	Yes
10^{-4}	4.79	No	4.43×10^{-2}	Yes
10^{-5}	4.79	No	4.85×10^{-2}	Yes
10^{-6}	4.79	No	4.99×10^{-2}	Yes

[a]The final objective is f_*. "KKT?" indicates whether the computed solution is a KKT point for the relaxed problem.

Relaxation of System-Level Constraints

One relaxation is to treat the system interdisciplinary consistency constraints as inequalities rather than strict equalities.[10,12] Tolerances on those inequalities should be as loose as possible to prevent breakdown of numerical optimization algorithms. At the same time one must continue to impose as tight a tolerance as possible on the convergence of the subsystem problems so that the system constraints and their sensitivities are properly evaluated.

This approach has limitations. If the system equality constraints are not satisfied, the system variables correspond to a design that is not physically consistent (because the multidisciplinary analysis relations are not satisfied) and that also violates the design constraints of one or more disciplines.

Furthermore, this relaxation is not guaranteed to relieve the computational difficulties or to lead to designs that are nearly optimal. This is illustrated by the CO_2 formulation of problem (10). We relax the system constraints to be

$$c_1(s, t_1, t_2) \le \varepsilon, \qquad c_2(s, t_1, t_2) \le \varepsilon$$

for $\varepsilon > 0$. The relaxed system problem is strictly convex and has a unique minimizer.

First suppose we begin at $(s, t_1, t_2) = (-3, -3, -3)$, which led to the false solution (17). The results for different values of ε are reported in the second and third columns of Table 2. In all cases NPSOL fails to find a legitimate solution for the relaxed system problem, even though the relaxed program is convex. The algorithm is undone by the singularity of the constraint Jacobian at feasible and realizable values of the system variables (i.e., rows of the Jacobian vanish and/or become linearly dependent).

On the other hand, if we repeat the same experiment starting at $(s, t_1, t_2) = (1, 1, 1)$, which allowed us to find the correct solution (16), we encounter another issue. The results are summarized in the fourth and fifth columns of Table 2. This time the relaxed constraints allow NPSOL to find objective values that are significantly better than the true value, 5.05×10^{-2}.

Whether such deviations are acceptable in a realistic problem depends on the application. If the objective function is a physical value, such as range of an aircraft, the deviation can be significant. By relaxing the system consistency tolerances, we are now sometimes able to solve the wrong problem more easily.

As this example makes clear, relaxing the system constraints does not necessarily repair CO_2 and can introduce another difficulty. It is not clear that there is an ideal relaxed tolerance for the system constraints that allows one to solve the problem while not distorting the solution to unacceptable levels. Furthermore, one would expect this sensitivity to relaxation to be greater in more complex problems, as computational tests demonstrate.[10] In general, one cannot know, a priori, the effect of such relaxations on the optimal solution because such knowledge would require a perturbation analysis at the optimal solution.

In our simple tests we varied the tolerances parametrically. In practical engineering problems users may have a physics-based or an engineering-based idea of acceptable variations in the design variables, objectives, or constraints. For example, variations of 1 kg may be acceptable in the gross liftoff weight for an aircraft. However, in order to translate an acceptable variation in a physical quantity to an acceptable tolerance for infeasibility of the system constraints

one would need an estimate of the associated Lagrange multipliers at an optimal solution, which is not available a priori.

Relaxation via Response Surface Methodology

Response surface methodology (RSM) has been proposed as another approach to relaxing CO.[31–33] In one technique the disciplinary analyses serve to build response surfaces that replace the analyses as function evaluators in the subproblems. This is a conventional use of RSM, and it does not alleviate the analytical difficulties of CO: the optimization problem structure remains unchanged (although the additional uncertainty of the quality of RSM approximation now contributes to the formulation).

The second approach uses RSM to build response surfaces that approximate the DF. A detailed analysis of this approach would depend both on the problem and on the specific type of data fitting surfaces used. As we are dealing with the fundamental CO formulation in this paper, such an analysis is not in its scope. However, the conceptual difficulty of the approach can be summarized as follows. Although response surfaces do ease the solution of the system problem by virtue of distancing the problem from the original CO formulation, it then becomes difficult to say what problem we are actually solving. In CO the use of response surfaces is mainly aimed at avoiding difficulties with CO, not difficulties with the MDO problem itself.

Alternative Optimization Algorithms

In this work we have relied on SQP algorithms as the means of solving the CO system problem because SQP methods are generally the most efficient for equality constrained optimization. As we have shown, CO has analytical features that hinder the successful operation of SQP algorithms. In particular, the singularity of the system constraint Jacobian causes difficulty for SQP algorithms. Moreover, projection into the null space of the system constraint Jacobians, on which SQP methods rely, is untenable and causes erratic computational behavior.

Thareja and Haftka[9] applied both a feasible direction and a penalty function algorithm to a version of CO_2. They also found that the designs at which their algorithms terminated were highly variable and very sensitive to the choice of parameters in the optimization algorithms. For the very simple problems presented here quadratic penalty methods applied to the CO system problem yield reasonably reliable, but not efficient, solutions.

Alternative: Distributed Analysis

We can formulate the MDO problem in a way that respects the requirements of conventional nonlinear programming analysis and algorithms and avoids the analytical difficulties of the bilevel approaches we have discussed. We call this approach Distributed Analysis Optimization (DAO). Instances of this class of formulations have appeared previously.[34–36]

DAO treats the implicit interdisciplinary consistency constraints in the MDA as explicit equality constraints in the optimization problem. FIO becomes

$$\begin{aligned} \min_{s,l_1,l_2,t_1,t_2} \quad & f(s,t_1,t_2) \\ \text{s.t.} \quad & g_1(s,l_1,t_1) \geq 0, \qquad t_1 = a_1(s,l_1,t_2) \\ & g_2(s,l_2,t_2) \geq 0, \qquad t_2 = a_2(s,l_1,t_1) \end{aligned}$$

where $a_1(s,l_1,t_2) = A_1(s,l_1,t_2)$, and $a_2(s,l_2,t_1) = A_2(s,l_2,t_1)$. DAO enjoys a measure of disciplinary autonomy in that the disciplinary analyses can be performed independently. Details may be elsewhere.[3,4]

The DAO formulation has the same smoothness and stability properties as FIO. There is no difficulty with Lagrange multipliers or nonsmoothness. Because the analytical properties of DAO are the same as those of FIO, performance of optimization algorithms applied to DAO will not suffer as a consequence of reformulation.

A tradeoff in DAO and CO is in the treatment of local design variables. The DAO system problem operates on the entire set of design variables, while the local variables are absent from the CO system problem. To address this limitation in DAO, one needs to take advantage of the block structure induced by the disciplines. On the other hand, the same computational elements (e.g., disciplinary analyses and sensitivities) are needed to implement FIO, DAO, and CO.

Lessons Learned

We have shown that CO gives rise to nonlinear programming problems that are difficult to solve by conventional optimization algorithms. The difficulty is caused not by intrinsic properties of the original MDO problem, but rather by the bilevel representation in CO. As computational experience suggests,[10,12] if a CO formulation can be solved at all it is unlikely that it will be solved quickly. Depending on the application, the lack of efficiency may figure in selecting a problem formulation. Requirements for inclusion of expensive, high-fidelity analyses can preclude the use of CO in a practical environment.[11] We comment on additional considerations that should assist a user in choosing a problem formulation.

Robustness

The concept of robustness in nonlinear programming has two major components. First, it denotes the ability of an algorithm to find an answer, starting from an arbitrary initial point. Second, a robust algorithm ensures that the answer found is correct and, if not, terminates with an informative message. The second feature is, arguably, by far the more important of the two, and it relies on the availability of tractable stationarity conditions. CO formulations are not well-posed from the perspective of conventional nonlinear programming because the system problems do not satisfy the standard optimality conditions.

An algorithm applied to the system problem in CO cannot be expected to exhibit robust behavior. Our analysis and examples show that CO, when started at or near solutions, will generally leave the region of feasible designs. The lack of a verifiable stopping criterion for CO also explains the tendency of NLP software to halt at points that are not solutions because progress cannot be made. To provide a reliable stopping criterion in terms of FIO, one would have to implement FIO, which would defeat the purpose of implementing CO in the first place. The same reasoning also prevents uses of a hybrid approach, e.g., CO combined with FIO.

We have also shown that attempts to relax CO may not alleviate the computational difficulties. Relaxation may not make it possible to find the solution reliably and can also lead to answers that appear significantly better than the correct solution but violate the physical consistency of the MDA and violate disciplinary constraints.

These features add up to the most serious drawback of CO—the lack of robustness in using conventional nonlinear programming algorithms. Again, this is caused by the nature of the system problem in CO, not by the nature of the original physical problem nor by deficiencies of standard optimization algorithms.

The use of quadratic penalty methods can be more robust (it was for the very simple problems presented here) but not efficient.

Dimensionality of the System Problem

Because the local design variables are eliminated from the system problem in CO, both the system and the subsystem problems have a reduced number of variables, compared to the total set of design variables. However, this reduction is achieved at the price of system problems that are difficult to solve.

Ease of Implementation and Execution

When a problem formulation requires an MDA capability, the effort is expended not just in implementing the MDA, but also during the execution of MDA because the processing of the disciplinary inputs may require extensive human intervention. CO is claimed to ease problem implementation and execution because an explicit MDA capability is not required. We believe this claim has not been proven satisfactorily as yet.

On the positive side, local disciplinary variables need not be treated as optimization variables at the system level. On the other hand, although MDA is not implemented in CO the flow of information among the disciplines still remains. That is, in our example of

aerostructural interaction structures still require input from aerodynamics, and conversely. This data exchange occupies a large portion of the implementation effort (regardless of the problem formulation) and adds to the execution effort. Because, typically, CO_2 would require many iterations to attain some level of convergence, human intervention to handle interdisciplinary inputs can be significant.

Moreover, because of the delinquent nature of the system problem in CO much time can be expended on fine tuning both the problem formulation and the optimization algorithm to produce answers.[10] For small test problems we found that implementing CO was more laborious than implementing FIO or DAO; however, small problems do not provide a fair test of the necessary implementation effort. Observations on the effort required to implement CO for more realistic problems can be found elsewhere.[10] This points to the need for careful measurement of labor expended on various parts of the problem implementation in MDO: the modeling, the problem statement, the optimization. Moreover, careful comparison must be made with the corresponding time expenditures when using, say, the standard formulation. Until such accounting is done on a realistic problem, substantiating claims on the increased ease of implementation will be difficult.

Conclusions

Bilevel approaches suggest themselves naturally as a decomposition strategy for large problems, and researchers have repeatedly turned to them in an attempt to deal with engineering systems in a decentralized fashion. Although computational efficiency is one of the goals of bilevel approaches to the optimization of complex, coupled systems, the computational inefficiency that often results in practice is viewed, first, as a feature that will be ameliorated by increases in available computing power and, second, as less significant than the conjectured benefits, such as ease of problem synthesis and implementation, disciplinary autonomy, or a problem decomposition that reflects certain organizational procedures.

The attempt to preserve disciplinary autonomy and reduce system complexity gives bilevel methods their intuitive appeal. However, to evaluate an approach to MDO one must answer a number of questions concerning the resulting optimization problem(s). For methods based on decomposition and disciplinary autonomy, what manner of autonomy is actually afforded? What are the analytical and computational advantages and disadvantages attendant upon disciplinary autonomy? Do the benefits that motivate the use of disciplinary autonomy, such as ease of implementation or computational efficiency, actually obtain in the resulting approach?

In this work we have examined in detail the analytical and computational features of a frequently proposed bilevel approach to MDO. The analytical features have a practical impact on the ability of nonlinear programming algorithms to solve the optimization problems that result from this approach. The study has illustrated the distinction between the intrinsic geometry of the feasible set and the way in which that set is represented in terms of constraints. Reformulated problems can introduce analytical features that cause difficulties for optimization algorithms.[29,30]

To illustrate the computational consequences of the analytical features of CO, we have conducted numerous tests on problems chosen for their simplicity to remove intrinsic difficulties of the test problems from the experiments. This means that the computational conditions of the tests were more benign than what can be realistically expected in practice. We also used analytical derivatives in the disciplinary subproblems and either highly accurate or analytical system sensitivities obtained from postoptimality sensitivity analysis of the disciplinary solutions. The numerical tests substantiated the analysis by revealing the following tendencies. Occasionally, a felicitous combination of optimization parameters and starting point would enable us to solve the CO system problem, although at considerably greater cost than FIO or DAO. However, the solution could not be accomplished reliably and efficiently. We observed that the solution was most reliably achieved when all of the system iterates were strictly nonrealizable (i.e., not realizable for all disciplines).

In summary, two characteristics of CO stand out. On the one hand, the approach does afford the user increased disciplinary autonomy. On the other, CO system problems are neither efficiently nor robustly soluble using conventional nonlinear programming algorithms. There is particular trouble at the points of interest, i.e., at or near realizable or interdisciplinary feasible points. Although one can devise algorithmic approaches that avoid these points, one still must be sure that the ultimate solution is realizable and hence physically meaningful for all disciplines. In selecting a problem formulation the user must weigh the relative importance of these features for the practical solution of the application problem in question.

Acknowledgments

Part of R. M. Lewis's research was supported by NASA under Contract NAS1-97046 while this author was in residence at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia. We wish to thank R. Haftka for bringing to our attention the antecedents of CO. Thanks are also given to Sean Wakayama for an example that suggested problem (9) and to J.-F. Barthelemy, S. Nash, and T. Zang for careful readings of the draft and many helpful comments.

References

[1] Alexandrov, N. M., and Lewis, R. M., "Comparative Properties of Collaborative Optimization and Other Approaches to MDO," *Engineering Design Optimization*, MCB Univ. Press, Bradford, England, U.K., 1999, pp. 39–46.

[2] Alexandrov, N. M., and Lewis, R. M., "Analytical and Computational Aspects of Collaborative Optimization," NASA/TM-210104-2000, April 2000.

[3] Alexandrov, N. M., and Lewis, R. M., "Analytical and Computational Properties of Distributed Approaches to MDO," *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* [CD-ROM], AIAA, Reston, VA, 2000.

[4] Alexandrov, N. M., and Lewis, R. M., "Algorithmic Perspectives on Problem Formulations in MDO," *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* [CD-ROM], AIAA, Reston, VA, 2000.

[5] Braun, R., "Collaborative Optimization: An Architecture for Large-Scale Distributed Design," Ph.D. Dissertation, Dept. of Aeronautics and Astronautics, Stanford Univ., Stanford, CA, May 1996.

[6] 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.

[7] 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. M. Alexandrov and M. Y. Hussaini, Society for Industrial and Applied Mathematics, Philadelphia, 1997, pp. 98–116.

[8] Sobieski, I., and Kroo, I., "Aircraft Design Using Collaborative Optimization, AIAA Paper 96-0715, Jan. 1996.

[9] Thareja, R., and Haftka, R. T., "Numerical Difficulties Associated with Using Equality Constraints to Achieve Multi-Level Decomposition in Structural Optimization," AIAA Paper 86-0854, 1986.

[10] Cormier, T., Scott, A., Ledsinger, L., McCormick, D., Way, D., and Olds, J., "Comparison of Collaborative Optimization to Conventional Design Techniques for Conceptual RLV," *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* [CD-ROM], AIAA, Reston, VA, 2000.

[11] Giesing, J. P., and Barthelemy, J. F., "A Summary of Industry MDO Applications and Needs," AIAA Paper 98-4737, Sept. 1998.

[12] Kodiyalam, S., "Evaluation of Methods for Multidisciplinary Design Optimization, Phase I," NASA CR-1998-208716, Sept. 1998.

[13] Balling, R. J., and Sobieszczanski-Sobieski, S., "Optimization of Coupled Systems: A Critical Overview of Approaches," *Proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA, Washington, DC, 1994, pp. 753–773.

[14] Balling, R. J., and Sobieszczanski-Sobieski, S., "An Algorithm for Solving the System-Level Problem in Multilevel Optimization," *Proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA, Washington, DC, 1994, pp. 794–809.

[15] Sobieszczanski-Sobieski, J., "A Linear Decomposition Method for Large Optimization Problems—Blueprint for Development," NASA TM-83248-1982, March 1982.

[16] Barthelemy, J.-F. M., "Development of a Multilevel Optimization Approach to the Design of Modern Engineering Systems," NASA CR-172184, Aug. 1983.

[17] Sobieszczanski-Sobieski, J., James, B., and Dovi, A., "Structural Optimization by Multilevel Decomposition," AIAA Paper 83-0832, May 1983.

[18] Sobieszczanski-Sobieski, J., James, B. B., and Riley, M. F., "Structural Sizing by Generalized, Multilevel Optimization," *AIAA Journal*, Vol. 25, No. 1, 1987, pp. 139–145.

[19]Sobieszczanski-Sobieski, J., "Two Alternative Ways for Solving the Coordination Problem in Multilevel Optimization," *Structural Optimization*, Vol. 6, No. 3, 1993, pp. 205–215.

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

[21]Schmit, L. A., Jr., and Mehrinfar, M., "Multilevel Optimum Design of Structures with Fiber-Composite Stiffened-Panel Components," *AIAA Journal*, Vol. 20, No. 1, 1982, pp. 138–147.

[22]Schmit, L. A., and Chang, K. J., "A Multilevel Method for Structural Synthesis," *A Collection of Technical Papers: AIAA/ASME/ASCE/AHS 25th Structures, Structural Dynamics, and Materials Conference*, AIAA, New York, 1984.

[23]Adelman, H. M., Walsh, J. L., and Pritchard, J. I., "Recent Advances in Integrated Multidisciplinary Optimization of Rotorcraft," AIAA Paper 92-4777, Sept. 1992.

[24]Walsh, J., Young, K., Pritchard, J., Adelman, H., and Mantay, W., "Integrated Aerodynamic/Dynamic/Structural Optimization of Helicopter Rotor Blades Using Multilevel Decomposition," NASA TP-3465, Jan. 1995.

[25]Alexandrov, N. M., and Kodiyalam, S., "Initial Results of an MDO Method Evaluation Study," *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA, Reston, VA, 1998, pp. 1315–1327.

[26]Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "User's Guide for NPSOL (Version 5.0): A Fortran Package for Nonlinear Programming," Stanford Univ., Stanford, CA, 1995.

[27]Barthelemy, J. F. M., and Sobieszczanski-Sobieski, J., "Extrapolation of Optimum Design Based on Sensitivity Derivatives," *AIAA Journal*, Vol. 21, No. 5, 1983, pp. 797–799.

[28]Barthelemy, J. F. M., and Riley, M. F., "Improved Multilevel Optimization Approach for the Design of Complex Engineering Systems," *AIAA Journal*, Vol. 26, No. 3, 1988, pp. 353–360.

[29]Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, London, 1981.

[30]Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "Software and Its Relationship to Methods," *Numerical Optimization 1984*, edited by P. T. Boggs, R. H. Byrd, and R. B. Schnabel, Society for Industrial and Applied Mathematics, Philadelphia, 1985, pp. 139–159.

[31]Manning, V. M., "High Speed Civil Transport Design Using Collaborative Optimization and Approximate Models," NASA Final Rept., Partial fulfillment of the research funded by Grant NCC-1-253, 1999.

[32]Sobieski, I. P., "Collaborative Optimization Using Response Surface Estimation," Ph.D. Dissertation, Stanford Univ., Stanford, CA, 1998.

[33]Sobieski, I. P., Manning, V. M., and Kroo, I. M., "Response Surface Estimation and Refinement in Collaborative Optimization," AIAA Paper 98-4753, 1998.

[34]Cramer, E. J., Dennis, J. E., Jr., Frank, P. D., Lewis, R. M., and Shubin, G. R., "On Alternative Problem Formulations for Multidisciplinary Design Optimization," *Proceedings of the 4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, AIAA, Washington, DC, 1992, pp. 518–530.

[35]Dennis, J. E., Jr., and Lewis, R. M., "A Comparison of Nonlinear Programming Approaches to an Elliptic Inverse Problem and a New Domain Decomposition Approach," Dept. of Computational and Applied Mathematics, Tech. Rept. TR94-33, Rice Univ., Houston, TX, Aug. 1994.

[36]Lewis, R. M., "Practical Aspects of Variable Reduction Formulations and Reduced Basis Algorithms in Multidisciplinary Design Optimization," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. Alexandrov and M. Y. Hussaini, Society for Industrial and Applied Mathematics, Philadelphia, 1997, pp. 172–188.

A. Messac
Associate Editor