IMPLEMENTATION AND PERFORMANCE ISSUES IN COLLABORATIVE OPTIMIZATION

Robert Braun*, Peter Gage†, Ilan Kroo‡, Ian Sobieski§

Abstract

Collaborative optimization is a multidisciplinary design architecture that is well-suited to large-scale multidisciplinary optimization problems. This paper compares this approach with other architectures, examines the details of the formulation, and some aspects of its performance. A particular version of the architecture is proposed to better accommodate the occurrence of multiple feasible regions. The use of system level inequality constraints is shown to increase the convergence rate. A series of simple test problems, demonstrated to challenge related optimization architectures, is successfully solved with collaborative optimization.

Nomenclature

c_i= local analysis constraint
F= objective function
g_{ij}= compatibility constraint
s= slack variable
$\overline{x}_i$= domain-specific design variable
x_i= interdisciplinary coupling variable
y_{ij}= coupling variable
z= system level design variable

Introduction

Many strategies for the optimization of multidisciplinary analysis problems have been proposed. Development of these optimization architectures is an active field of research, strongly coupled with decomposition methodology [1, 2, 3]. Selection of the appropriate optimization architecture is of paramount importance in the efficient solution of multidisciplinary analysis problems. In some cases, one architecture may yield the solution quickly; whereas, another may be slow or even non-convergent [4].

Collaborative optimization (CO) is a design architecture which preserves traditional disciplinary groupings by allowing parallel development of the design. A coordination problem ensures that the parallel design/analysis problems converge to a single compatible system. In this paper a theoretical description of the architecture is followed by a description of a specific calculus-based implementation.

Development of collaborative optimization is discussed in [2, 5, 6]. The algorithm has been applied by researchers to a number of different design problems; the trajectory of a lunar ascent vehicle [5, 6], the sizing of a medium range transport aircraft [7], and the design of a single-stage-to-orbit booster [5, 8]. In all of these problems the algorithm successfully converged to an optimal solution. In this paper CO is applied to a set of sample problems that have been shown to cause difficulty for related architectures [9]. Though the successful solution of these problems does not constitute a general convergence proof, it does increase confidence in the practical applicability of the algorithm.

The decomposition framework implicit in the calculus-based expression of the collaborative architecture can, in some cases, introduce multiple subspace solution regions. A simple problem is used to demonstrate how these disjoint regions can cause discontinuities in the system level design space leading to difficulties for gradient based optimization. One strategy for avoiding this difficulty involves reformulating the system level optimization problem through the introduction of an extra variable and constraint. This method is described and evaluated in this paper.

The optimizer used at the system level is a sequential programming method (SQP). This algorithm uses a linear model for the constraint set. Because of this, changing the nonlinear compatibility con-

*Aerospace Engineer, NASA-LaRC, Member AIAA
†Lecturer, Australian Defense Academy, Member AIAA
‡Assoc. Professor,Stanford University, Member AIAA
§Graduate Student, Stanford University, Member AIAA

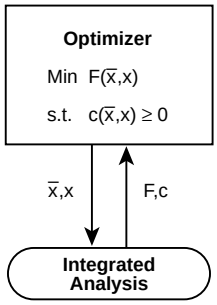


Figure 1: Standard optimization approach..

straint from an equality to an inequality constraint is shown to improve convergence performance.

Optimization Approaches

To provide a basis for comparison with the collaborative architecture this section describes several optimization architectures.

Standard Approach

Because the formulation illustrated in Fig. 1 has the broadest use in aerospace design, and is therefore the standard optimization approach. For a multidisciplinary system, use of this approach requires an integrated set of analysis models such that for a given set of design variables ($X = [\overline{x}_i \; x_i]$), the analysis returns the values of the constraints (c) and the objective function (F).

In this approach, interdisciplinary coupling is accommodated within the integrated set of analyses as part of each overall function evaluation. In order for the integrated analyses to achieve a design that is feasible, this approach typically requires satisfaction of an iterative process during each complete analysis call.

Consider the structural and aerodynamic design of a minimum drag wing. The design variables may include the wing span, area, sweep, twist, and taper; the constraints may include wing weight and total lift. In the standard approach [10], an optimizer passes values of the design variables to the integrated set of structural, weight estimation, and aerodynamic analyses. These analyses must be iteratively evaluated until they converge on a consistent (i.e. multidisciplinary feasible) solution. The drag and constraint values are then used by the optimizer to modify the design variables to improve the design.

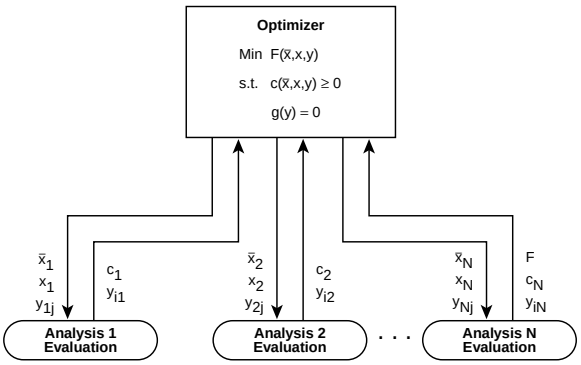


Figure 2: All-at-once optimization architecture.

Distributed Analysis

Distributed architectures, like the all-at-once and disciplinary feasible constraint methods detailed later, have numerous organizational and computational advantages [11, 12, 13, 14] over the standard approach. Organizational advantages include: (1) a more natural fit to the current disciplinary expertise structure found in most design organizations, (2) empowerment of the disciplinary experts in the design decision process (through subspace optimization) and (3) the flexibility to efficiently alter a portion of the design analyses without having to repose the complete problem. Similarly, computational advantages of a distributed system include: (1) a reduction in the integration and communication requirements, (2) a parallel optimization architecture which is readily operable on a suite of heterogeneous platforms, (3) removal of iteration loops (resulting in a smoother design space) and (4) a reduced level of disciplinary sequencing.

All-At-Once

Research in decomposition analysis has led to a class of alternative formulations known by several names: simultaneous analysis and design [4, 15], all-at-once [12], or optimizer-based decomposition [2, 3, 16]. As sketched in Fig. 2, this formulation allows the N analysis-blocks to be executed in parallel.

Here, each analysis-block is responsible for evaluating its own set of the originally partitioned constraints, c_i. Furthermore, for each coupling variable, y_{ij} shown in Fig. 2, an equality constraint and a design variable are added to the optimization problem set to enforce interdisciplinary compatibility [16, 17]. When satisfied, these equality constraints, g_{ij}, require that the value of a variable computed in analysis block j match the value of the

equivalent variable input to analysis block i. Within the optimization problem, these coupling variables, y_{ij}, are included in the design variable vector along with both the disciplinary, $\overline{x}_i$, and interdisciplinary inputs, x_i.

In comparison to the standard formulation the requirement of producing a compatible multidisciplinary model is removed from the analyses, becoming an additional task of optimization. In this manner, interdisciplinary feasibility (among the parallel analysis-blocks) is only required at the solution, where $g_{ij} = 0$. Unlike the standard approach, neither the domain-specific nor interdisciplinary constraints are solved within the analysis-blocks. Instead, these function evaluators provide residual information to the system-level optimizer which has the responsibility of constraint satisfaction. In some cases, these formulative changes have been shown to produce a smoother design space than the standard approach as well as computational savings through the removal of implicit iteration loops from the original analysis block [17, 16, 15, 18].

Although the approach of Fig. 2 may be computationally faster than the use of the standard optimization approach it retains some of the standard approaches' shortcomings. The analysis groups are still removed from the design decision-process, acting simply as function evaluators. Additionally, in large-scale applications, with thousands of design variables and constraints, the use of a single optimizer may lead to numerous problems including a significant increase in the communication requirements between the analysis blocks and optimizer. For a tightly-coupled problem, the all-at-once feasibility characteristics of this approach may also lead to a large increase in the number of auxiliary design variables and constraints, since all design decisions (no matter how localized) are made by the single optimization routine.

Discipline Feasible Constraint

Disciplinary feasible constraint methods allow for greater design authority among the distributed subproblems and are less sensitive to increasing problem size. In these methods the individual analysis blocks are required to return a design candidate that satisfies the disciplinary governing equations as well as the domain-specific constraints, $c(\overline{x})$, but may not be interdisciplinary compatible. A system-level optimizer is utilized to ensure interdisciplinary compatibility at the overall solution and minimize the objective function.

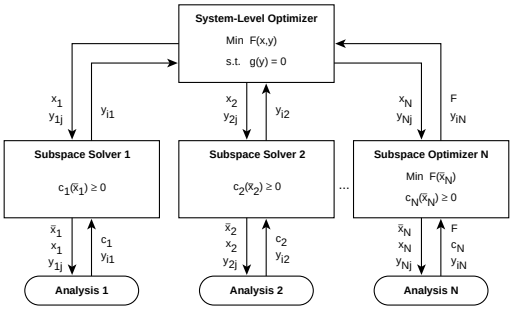


Figure 3: Disciplinary feasible constraint optimization architecture.

CSSO

An example of the this approach is Concurrent Subspace Optimization (CSSO) in which multiple subspace optimization problems are driven by a system-level optimizer that provides overall coordination. This approach to distributed design was first introduced in Ref. [19]. Variations of the basic formulation are presented in Refs. [20, 21, 22]. In CSSO, approximations of the other analysis groups' constraints are included in each subspace optimization problem set. The subproblems also partition the responsibility of constraint satisfaction. While this approach has been successfully applied to numerous design problems [21, 22], convergence difficulties are reported in Ref. [9] on the simple example of the following section.

Collaborative Optimization

Collaborative optimization is another discipline feasible constraint method. The collaborative optimization architecture is designed to promote disciplinary autonomy while achieving interdisciplinary compatibility. As sketched in Fig. 4, the problem is decomposed along analysis-convenient boundaries and subspace optimizers are integrated with each analysis-block. Through subspace optimization each group is given control over its own set of local design variables and is charged with satisfying its own domain-specific constraints. Explicit knowledge of the other groups' constraints or design variables is not required. The objective of each subspace optimizer is to agree upon the values of the interdisciplinary variables with the other groups. A system-level optimizer is employed to coordinate this process while minimizing the overall objective.

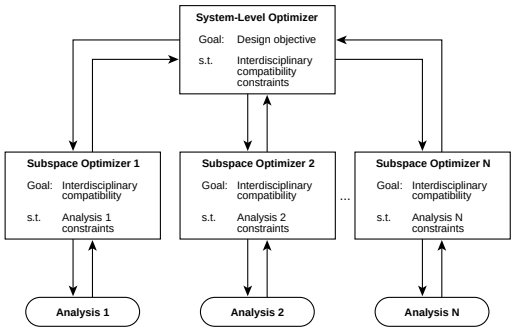


Figure 4: The basic collaborative optimization architecture.

The fundamental idea behind the development of the collaborative optimization architecture is that disciplinary experts should participate in the design decision process while not having to fully address local changes imposed by the other groups of the system. This decentralized decision strategy is not only a practical approach to design, but may also allow for the use of existing disciplinary analyses without major modification. This is not a trivial advantage, as the practical acceptance of many MDO techniques is limited by their implementation overhead requirements [5].

As sketched in Fig. 4, the system-level optimizer relies on information provided by repeated subspace optimization to coordinate the various subproblem optimizations. One means to convey this information, as discussed in Ref. [5, 23] and used later in this paper, is with gradients. However, alternative methods may also be possible in which the system-level optimization algorithm is not restricted to gradient-based techniques. Although beyond the scope of the present investigation, one can envision a conflict resolution strategy (analogous to an auction) in which the subspaces bid for desired changes in each interdisciplinary variables [24, 25, 26]. Here, each subspace may have a fixed allocation of points to spend in bidding for the interdisciplinary variables at each system-level iteration. In this manner, a group's strong convictions on the proper value of a certain interdisciplinary variable would be weighted appropriately.

Relative to CSSO, the collaborative optimization architecture provides a higher degree of design freedom within the subspaces while reducing the interdisciplinary communication requirements.

Characteristics	MDF	DFC	AAO
Integration cost	High	Low	Low
Modifications cost	Low	Medium	High
Num of Optimizations	One	Multiple	One
Problem size	Small	Medium	Large
Problem sparsity	Dense	Moderate	Sparse
Computational cost	High	Medium	Low
Overall performance	Slow	Medium	Fast
Robustness	Low	Medium	High

Table 1: Multidisciplinary optimization architecture comparison.

Comparative Characteristics

General characteristics of these three approaches to optimization (multidisciplinary feasible, MDF, disciplinary constraint feasible, DFC, and all-at-once, AAO) are tabulated for comparison in Table 1.

In aerospace design, most optimization studies performed today rely on a multidisciplinary feasible method. While this brute-force approach may be the simplest to comprehend, Table 1 list several performance drawbacks. On the other hand, while an all-at-once approach may provide the most efficient means toward a solution, this solution strategy may be difficult to implement, has large communications requirements, and leaves the domain-specific analyses with the role of function evaluation only. By maintaining several of the advantages of the all-at-once approach, while incorporating subspace optimization to augment the role of the disciplinary expert, the disciplinary constraint feasible methods have the most to offer the engineering community towards the solution of large-scale, practical design problems.

CO Performance Aspects

The collaborative architecture has been used by researchers to solve a number of different design problems [7, 8, 5]. Of the multi-level architectures only [27] has been formally proven to be convergent. Lacking such proof, confidence in the robustness of the collaborative approach is developed by examining the performance of the method on problems. Of particular interest here are the battery of simple quadratic problems that have caused difficulty for other discipline constraint feasible methods [9]. In the following sections, the results of the successful solution of these problems using CO is reported.

Application to Quadratic Test Problems

The first example problem is taken from Ref. [9] with two design variables, and a quadratic objective. The two constraints are linear, and the parameter β is used to modify their slopes, thereby shifting the location of the constrained optimum. Five different starting points are used to ensure that convergence is consistent.

$$\begin{aligned} \min_{X_1,X_2} \; & F = {X_1}^2 + {X_2}^2 \qquad (1) \\ s.t. \; & c_1 = X_1 + \beta X_2 < 4 \\ & c_2 = \beta X_1 + X_2 > 4 \\ & l_1 < X_1 < u_1 \\ & l_2 < X_2 < u_2 \end{aligned}$$

This example was used by Shankar to demonstrate that considerable effort is required to achieve convergence using the nonhierarchical decomposition strategy of Ref. [21]. In that architecture, constraints from each subproblem are modeled in the other subproblems, using a single Kreisselmeier-Steinhauser cumulative constraint. Coefficients are introduced to assign the fractional responsibility of subspace i in reducing the violation of the constraints from subspace j. Inappropriate selection of the values for these coefficients caused termination at non-optimal points.

Example Problem 1: Formulation

The collaborative formulation for this problem is shown in Fig. 5. This formulation is one specific expression of the more general collaborative framework described in the previous section. Here a quadratic discrepancy function consisting of the local and target values of the various design variables and parameters serves as the subproblem objective function. The system level optimization is responsible for minimizing the objective value while adjusting the target variable values to obtain a multi-disciplinary feasible design. Each change in the system level design variable set (ie. the target values) requires a full optimizations at the sub-problem level. This process is made more efficient through the use of analytic gradients for the compatibility constraints, obtained through post-optimality analysis [23].

At each iteration, all system variables are sent to each subspace. The subspaces' objective is to reduce the discrepancy between these system-level target variable values and the value of corresponding local constraint variables. The subspace solution need not necessarily match these system levels targets, but local feasibility must be maintained. The system is responsible for minimizing the objective while adjusting the system-level design variable values to obtain a multidisciplinary feasible design.

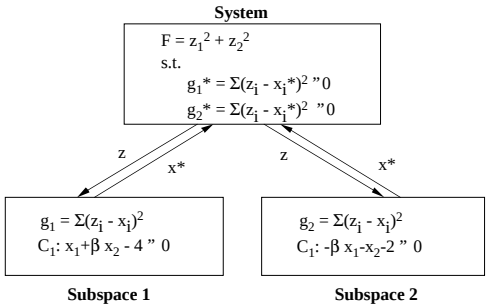


Figure 5: Collaborative formulation for Example 1

β	Starting point					Solution
	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	
0.0	**C 17**	**C 17**	**C 17**	**C 16**	**C 18**	(0.0,2.0)
0.1	**C 17**	**C 17**	**C 17**	**C 15**	**C 21**	(0.198,1.98)
0.3	**C 19**	**C 17**	**C 17**	**C 15**	**C 14**	(0.55,1.85)
0.5	**C 19**	**C 15**	**C 17**	**C 12**	**C 19**	(0.8,1.6)
1.0	**C 18**	**C 17**	**C 17**	**C 17**	**C 16**	(1.0,1.0)

Table 2: Optimizer termination code and number of system-level iterations, for different and for several starting points

Example 1: Results

Successful convergence was achieved for all values of β, and all starting points, as shown in Table 2. The optimizer termination code denotes convergence by 'C'. The numbers of system-level iterations are shown next to the termination code for each optimization case. They compare favorably with nonhierarchical decomposition, which used up to 71 iterations.

Example 2: Formulation

The objective function of the second sample problem, also taken from [9], is the weighted sum of squares of the six design variables. There are six local subproblem constraints. Again, a parameter β is

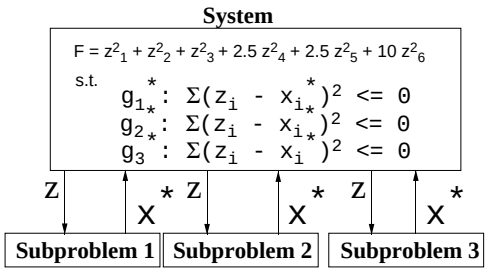


Figure 6: Collaborative formulation for Example 2.

used to modify the slope of these linear constraints to alter the location of the constrained optimum. Five different starting points are used to ensure that convergence is consistent.

The collaborative formulation decomposes the problem into the three subspaces as shown in Figure 6. There are three disciplinary constraints in the first subproblem, two in the second, and one in the third. The system objective is calculated at the system-level, with three compatibility constraints: one for each subproblem.

Example 2: Results

In Reference [9], Shankar et al. were unable to develop a combination of weighting coefficients to permit satisfactory convergence when the coupling between subspaces was strong (when the value of the parameter β was large). The best results from that study are reproduced in Table 3, in small font. The optimizer termination code 'WC' indicates that the algorithm converged at a non-optimal point. The results using collaborative optimization are also included in Table 3, in larger font and underlined. When there is no coupling between subproblems (Row 1, $\beta = 0$), the methods have similar performance. For weak coupling, the nonhierarchical decomposition algorithm can usually converge, but it takes many more iterations than collaborative optimization. Even for strong coupling, the collaborative approach converges successfully, with no increase in the number of iterations required.

The ability of CO to solve more complex design problems has been demonstrated in Refs. [6, 8, 7]. Here, however, the algorithm has been used to successfully solve a battery of problems known to cause trouble for other constraint feasible methods.

β	Starting point					Solution
	(0,0,0 0,0,0)	(1,2,3, -1,1,5)	(-10,4,4, 0.8,0.1,1)	(1,1,1, 1,1,1)	(-4,2,2, 0,1,1)	
0.0	C 17 **C 19**	C 17 **C 22**	C 19 **C 21**	C 17 **C 19**	C 17 **C 22**	(0.6,0.6,0.6, -2.0,-2.0,6.0)
0.1	C 110 **C 22**	C 190 **C 22**	C 113 **C 21**	C 72 **C 22**	WC 109 **C 22**	(-2.4,-2.4,7.0, -1.7,-1.8,4.8)
0.3	WC 105 **C 22**	C 203 **C 20**	C 103 **C 22**	C 163 **C 21**	WC 103 **C 21**	(-2.7,-2.7,8.0, -1.5,-1.8,1.9)
0.5	WC 103 **C 21**	WC 103 **C 19**	WC 104 **C 21**	WC 107 **C 21**	WC 107 **C 22**	(-1.7,-1.7,6.3, -1.5,-1.9,1.0)
1.0	WC 103 **C 21**	WC 103 **C 21**	WC 104 **C 20**	WC 102 **C 20**	WC 102 **C 20**	(-0.5,-0.5,4.2, -1.2.-2.0,0.7)

Table 3: Optimizer termination code and number of system-level iterations–Results for collaborative optimization are underlined

Multiple Solution Difficulty Posed by Nonlinear Subspaces

In general, a nonlinear programming problem may have multiple feasible regions. When solving such a problem, calculus-based optimization approaches typically guarantee convergence to a local solution [28, 29]. For the collaborative architecture, where the original-problem constraints are satisfied at the subspace level, the presence of multiple solution regions poses an additional difficulty. In the collaborative approach, system-level variable perturbations translate directly into subspace parameter variations. As the system optimizer converges upon a solution, the subspace parameters may vary over a relatively large range. As a result, the subspace optimizer may discontinuously jump from one feasible region to another. This subspace solution inconsistency may cause non-smoothness in the subspace objective function, thereby producing an erroneous system-level Jacobian and hampering system-level performance.

Demonstration: Constrained Rosenbrock Valley Function

As an example, consider augmenting the Rosenbrock valley function with a single nonlinear constraint. When solved with the collaborative architecture this constraint becomes part of the following subspace optimization problem.

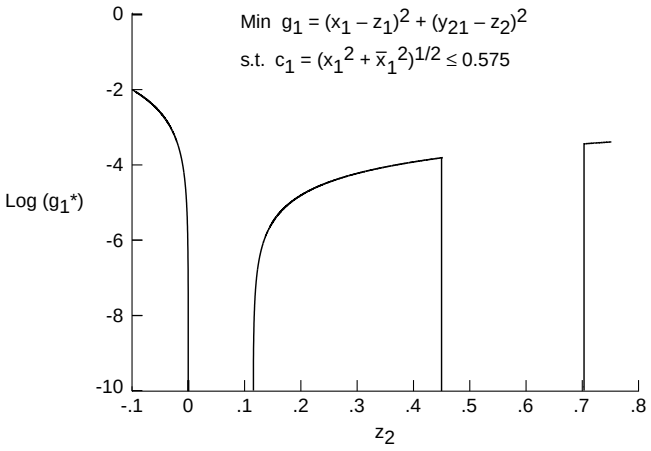


Figure 7: Optimal solution (g_1^*) to the constrained Rosenbrock valley subspace 1 design problem for various values of z_2; $z_1 = 0.5$.

$$\begin{aligned} \min_{\overline{x}_1, x_1} \quad & g_1 = (x_1 - z_1)^2 + (y_{12} - z_2)^2 \qquad (2) \\ s.t. \quad & c_1 = (x_1^2 + \overline{x}_1^2)^{\frac{1}{2}} \le 0.575 \\ & 0 \le x_1 \le 1 \\ & 0 \le \overline{x}_1 \le 1 \end{aligned}$$

where,

$$y_{12} = 100(\overline{x}_1 - x_1^2)^2 \qquad (3)$$

and the system-level targets have the following bounds,

$$-1 \le z_1 \le 1 \qquad (4)$$

$$-1 \le z_2 \le 1 \qquad (5)$$

To obtain a collaborative solution to this problem, the system-level optimizer will repeatedly send values of the system-level targets, z, to the subspaces. For each set of system-level targets, the subspaces return a solution g_i^* and a row of the system-level Jacobian. For simplicity, let us restrict our attention to the following case,

$$z_1 = 0.5 \qquad (6)$$

$$-1 \le z_2 \le 1 \qquad (7)$$

Repeatedly solving this subspace optimization problem (starting from the same initial point) for various values of z_2 yields the profiles of g_1^* and x_1^* depicted in Fig. 7.

Figure 7 shows two disconnected regions in which g_1^* is considered to be 0 ($\log g_1^* \le -10$). For these cases, no constraints are binding at the solution. The optimal objective function varies continuously beyond both sides of the first of these regions,

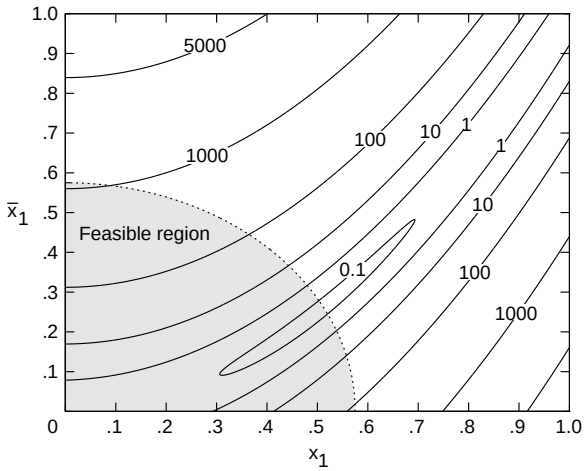


Figure 8: Constrained Rosenbrock valley subspace 1 design space; $z_1 = 0.5$, $z_2 = -0.25$.

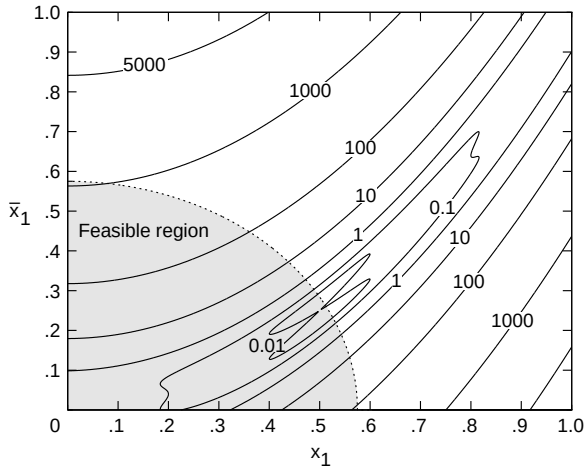


Figure 9: Constrained Rosenbrock valley subspace 1 design space; $z_1 = 0.5$, $z_2 = 0.1$.

$0 \le z_2 \le 0.1152$; however, the variation on either side of the second region, $0.45 \le z_2 \le 0.71$ is discontinuous. This discontinuous behavior, which is an indication of multiple solution regions, may hamper the computation of an accurate system-level search direction regardless of how this search direction is estimated.

To further illustrate this problem, the variation in the design space for a related sequence of subspace problems is illustrated in Figs. 8 to 10.

In each of these figures, which depict the subspace 1 design space for various values of z_2, the feasible region is shaded. These figures depict the subspace 1 design space for $z_2 = [-.25, 0, .05, .1, .25, .5, .95]$. The contours in each figure represent constant values of the objective function, g_1^*.

Clearly, the presence of multiple solution regions is dependent on the value of the system-level target,

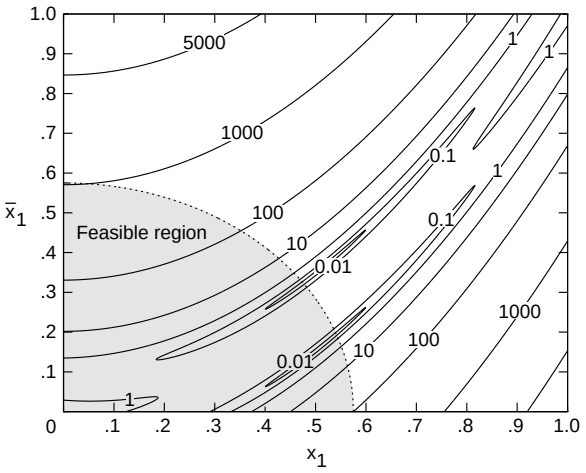


Figure 10: Constrained Rosenbrock valley subspace 1 design space; $z_1 = 0.5$, $z_2 = 0.95$.

z_2. For $z_2 < 0$, the subspace is characterized by a unique solution region with $g_1^* > 0$; whereas, for $0 < z_2 < 0.1152$, two feasible solution regions with $g_1^* = 0$ exist. For $z_2 \geq 0.1152$, one constrained ($g_1^* > 0$) and one unconstrained ($g_1^* = 0$) solution region are found.

Note that if this subspace problem had been solved just once (as in traditional methods), this feature may not have been of significance—a local solution would have been achieved. However, because the collaborative architecture relies on repeated solution of a related sequence of subspace problems, the existence of multiple solution regions is an important concern.

The presence of multiple solution regions is not caused by the introduction of a nonlinear constraint within this subspace. Rather, the nonlinearity of the interdisciplinary output, y_{12} is the cause of this ambiguity. This is evident through examination of the unconstrained optimization problem,

$$\min_{\overline{x}_1, x_1} \quad g_1 = (x_1 - z_1)^2 + (y_{12} - z_2)^2 \tag{8}$$
$$0 \leq x_1 \leq 1$$
$$0 \leq \overline{x}_1 \leq 1$$

where,

$$y_{12} = 100(\overline{x}_1 - x_1^2)^2 \tag{9}$$

The solution to this problem (for $z_2 > 0$) is,

$$g_1^* = 0 \tag{10}$$
$$x_1^* = z_1 \tag{11}$$
$$\overline{x}_1^* = z_1^2 + (\frac{z_2}{100})^{\frac{1}{2}} \tag{12}$$

Because eq. (16) has two roots (for $z_2 > 0$), two solution regions are created. This characteristic is a general problem which the collaborative architecture must overcome since many situations may arise in which nonlinear interdisciplinary outputs take a form similar to y_{12}. While the existence of these multiple solution regions cannot be generally removed, it is possible to repose the system level optimization problem to avoid this difficulty. Techniques which accomplish this subspace solution consistency are the subject of the following section.

Ensuring a Consistent Subspace Solution

This difficulty is avoided if each subspace optimization process is limited to the domain of a single solution region. This restriction, which is analogous to the local convergence restriction of standard calculus-based optimizers, may be enforced in several ways. One means to this end is to restart each subspace optimizer from the previously obtained solution. While this approach does not guarantee repeated convergence to the same local minimum, it has been found to help.

A second possibility is to add a small penalty-term to the subspace objective functions. Hence, the subspace objective function may be reposed as,

$$g_i(x) = \sum_{j=1}^{h_i'} (x_{ij} - z_{ij})^2 + \sum_{j=1+h_i'}^{h_i} (y_{ij} - z_{ij})^2 + \epsilon \sum_{j=1}^{h_i'} (x_{ij} - X_{ij}^*)^2 \tag{13}$$

where,

X^*= optimum subspace design variable vector from previous subspace call

ϵ= penalty magnitude

Unfortunately, in this proximal solution approach, the choice of ϵ will affect problem convergence. A third approach, applicable to constrained subspaces, is to replace the inequality constraints with slack-variables and equality constraints as,

$$c_i(\overline{x}, x) \geq 0 \quad \longrightarrow \quad \begin{cases} c_i(\overline{x}, x) - s_i & = 0 \\ s_i & \geq 0 \end{cases} \quad (14)$$

This approach does not eliminate the inequality; it is still present as a bound on the auxiliary domain-specific design-variable, s_i. Furthermore, the subspace problem size is increased. However, for active set methods the effect of this variable may not be seen in the determination of the search direction. In the present research, NPSOL [30] is utilized to perform subspace optimization. Within NPSOL, the search direction is obtained from solution of a quadratic programming subproblem of the form,

$$\min_{p} \quad \frac{\partial g}{\partial X}^T p + \frac{1}{2} p^T H p \quad (15)$$

$$s.t. \qquad l \leq Ap \leq u \quad (16)$$

where,
g: subspace objective
p: search direction
X: subspace design variables, $[x\ \overline{x}\ s_i]$
H: Hessian of the Lagrangian function
A: constraint Jacobian

In this case, since s_i appears linearly within the constraint set, its impact is not modeled within H. Therefore, as the subspace iterations proceed, the search directions computed by the algorithm are biased against varying s_i.

As a result, this slack-variable formulation refinement may be used to bias the subspace optimizer toward a particular solution region (e.g., where the bound is always active). In this manner, a smooth variation in the subspace objective function with respect to parameter variations is more likely, even in the presence of multiple solution regions.

To demonstrate this phenomena, the constrained Rosenbrock valley problem was reposed with a slack variable. In this case, repeated solution of the subspace optimization problem (starting from the same initial point) for all values of z_2 yields the g_1^* and x_1^* profiles depicted in Fig. 11.

For these solutions, the slack variable profile is shown in Fig. 12.

The initial guess for the slack variable was $s = 0$ (an active bound). Note that for all the cases in which an active and an inactive-constraint solution region were previously possible ($z_2 \geq 0.1152$), the active-constraint solution is now found, $s^* = 0$. Hence, this subspace solution consistency eliminates the discontinuity present in Fig. 7 yielding smooth

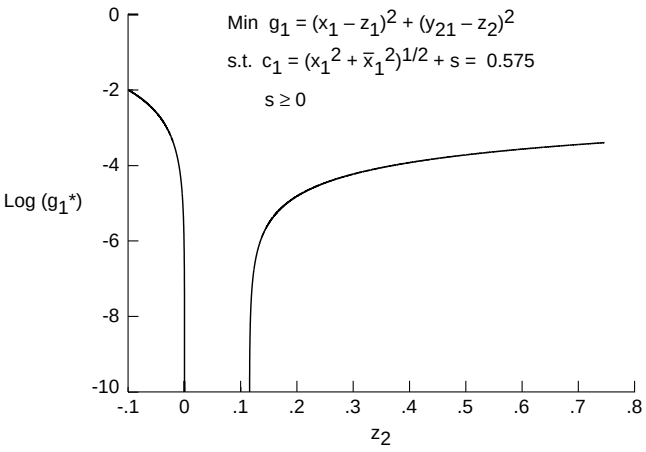


Figure 11: Optimal solution (g_1^*) to the constrained Rosenbrock valley Subspace 1 design problem for various values of z_2 with slack variable refinement; $z_1 = 0.5 (Compare with Figure 7.$

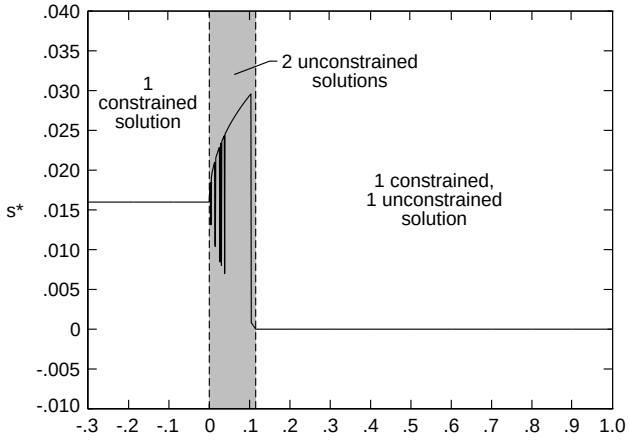
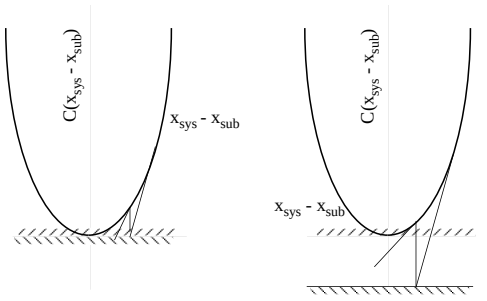


Figure 12: Optimal solution (s^*) to the constrained Rosenbrock valley Subspace 1 design problem for various values of z_2 with slack variable refinement; $z_1 = 0.5$.



(a) Equality constraint. Max step for linearized constraint is half what it should be.

(b) Inequality constraint. Longer steps possible, and constraint can become inactive.

Figure 13: Effective of inequality constraint on step size.

system-level constraint gradients. Note that in the region in which two inactive-constraint solutions were previously possible ($0 < z_2 < 0.1152$), this refinement has no effect and the subspace optimizer still jumps between solution regions. The use of this slack-variable approach to encourage a consistent subspace solution is demonstrated further in the highly constrained applications presented in Refs. [5, 8, 2].

Performance Issues

Effect of System-level Constraint Implementation

One reason for unnecessary system-level iterations is related to the details of the system-level constraint formulation. Convergence is not achieved for the problem of Figure 5, when the constraint in Subspace 1 is initially violated by the system values of the multidisciplinary variables. In such situations, the system optimizer fails to converge to a correct point, because this subspace constraint never becomes inactive. This is due to the line search algorithm of the SQP method in NPSOL [30]. The SQP method uses a linear approximation of the constraint, and the line search chooses a maximum step size that will not exceed the bound of the linearized constraint. In this sample problem, the step is too short to satisfy the true quadratic constraint, as shown in Figure 13.

This difficulty is resolved by modifying the system-level constraints to be inequalities, as shown in Figure 13. The linearized constraint is thus permitted to take negative values, although this is impossible for the quadratic constraint, and longer steps can be taken. The system variables can move beyond the values suggested by the local optimizer, and the local constraint can become appropriately inactive space. With this adjustment, which does not influence the generality of the method, the method converges as was shown in Table 2.

Conclusions

Collaborative optimization has been used to solve several distributed design problems. This work demonstrates the successful convergence of a class of problems that have proven problematic for other methods. While this empirical result is not general it does demonstrate that non-local constraint information may be accommodated in the collaborative framework, even for problems designed to challenge such methods. The creation of multiple subproblem solution regions due to the collaborative decomposition was identified. Three solutions for this problem were suggested. An implementation of a slack variable approach shows how the system-level optimizer may be biased to maintain a design point in one region. Finally, the use of linear models of the quadratic compatibility constraints by the system level optimizer is shown to produce slow convergence. The source of this slow convergence has been identified and the method improved by posing system-level equality constraints as inequalities.

References

[1] J. Sobieszczanski-Sobieski and R.T. Haftka. Multidisciplinary aerospace design optimization: Survey of recent developments. AIAA Paper 96-0711, Jan. 1996.

[2] I. Kroo. Decomposition and collaborative optimization for large-scale aerospace design. In *Multidisciplinary Design Optimization: State of the Art*, SIAM Publications, 1995.

[3] S.S. Altus, I.M. Kroo, and P.J. Gage. A genetic algorithm for scheduling and decomposition of multidisciplinary design problems. ASME Paper 95-141, Boston, MA, Sept. 1995.

[4] R.T. Haftka, J. Sobieszczanski-Sobieski, and S.L. Padula. On options for interdisciplinary analysis and design. *Structural Optimization*, 4:65–74, 1992.

[5] R.D. Braun. *Collaborative Optimization: An Architecture for Large-Scale Distributed Design*. PhD thesis, Stanford University, June 1996.

[6] R.D. Braun and I.M. Kroo. Development and application of the collaborative optimization architecture in a multidisciplinary design environment. SIAM, 1996.

[7] I. Sobieski and I. Kroo. Collaborative optimization applied to an aircraft design problem. AIAA Paper 96-0715, Jan. 1996.

[8] R. Braun, I. Kroo, and A. Moore. Use of the collaborative optimization architecture for launch vehicle design. AIAA Paper 96-4018, Bellevue, WA, Sept. 1996.

[9] J. Shankar, C. Ribbens, R. Haftka, and L. Watson. Computational study of nonhierarchical decomposition algorithm. *Computational Optimization and Applications*, 2:273–293, 1993.

[10] S. Wakayama and I. Kroo. Subsonic wing design using multidisciplinary optimization. AIAA Paper 94-4409, Panama City, FL, Sept. 1994.

[11] N. Alexandrov and Y. Hussaini, editors. *Multidisciplinary Design Optimization: State of the Art*. SIAM Publications, 1995.

[12] E.J. Cramer, J.E. Dennis, Jr., P.D. Frank, R.M. Lewis, and G.R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal of Optimization*, 4(4):754–776, Nov. 1994.

[13] T. Wagner. *A General Decomposition Methodology for Optimal System Design*. PhD thesis, University of Michigan, 1993.

[14] J.F. Barthelemy. Engineering applications of heuristic multilevel optimization methods. NASA CP-3031, Hampton VA, Sept. 1988.

[15] R.T. Haftka. Simultaneous analysis and design. *AIAA Journal*, 23(7):1099–1103, 1982.

[16] I. Kroo, S. Altus, R. Braun, P. Gage, and I. Sobieski. Multidisciplinary optimization methods for aircraft preliminary design. AIAA Paper 94-4325, Panama City, FL, Sept. 1994.

[17] P.J. Gage. *New Approaches to Optimization in Aerospace Conceptual Design*. PhD thesis, Stanford University, 1995.

[18] R.D. Braun, R.W. Powell, R.A. Lepsch, D.O. Stanley, and I.M. Kroo. Comparison of two multidisciplinary optimization strategies for launch vehicle design. *Journal of Spacecraft & Rockets*, 32(2):404–410, Mar.-Apr. 1995.

[19] J. Sobieszczanski-Sobieski. Optimization by decomposition: A step from hierarchic to non-hierarchic systems. In *Recent Advances in Multidisciplinary Analysis and Optimization*, NASA CP-3031, Hampton, VA, Sept. 1988.

[20] J.E. Renaud and G.A. Gabriele. Approximation in non-hierarchic system optimization. *AIAA Journal*, 32(1), Jan. 1994.

[21] E.D. Eason and J.E. Wright. Implementation of non-hierarchic decomposition for multidisciplinary system optimization. AIAA Paper 92-4822, Cleveland, OH, Sept. 1992.

[22] R.S. Sellar, S.M. Batill, and J.E. Renaud. Response surface based, concurrent subspace optimization for multidisciplinary system design. AIAA Paper 96-0714, Jan. 1996.

[23] R.D. Braun, P.J. Gage, and I.M. Kroo. Post-optimality analysis in aerospace vehicle design. AIAA Paper 93-3932, Aug. 1993.

[24] S.P. Hargreaves et al. *Game Theory : A Critical Introduction*. Routledge Publications, 1995.

[25] F.C. Zagare. *Game Theory : Concepts and Applications*. Sage Publications, 1984.

[26] W.F. Lucas, editor. *Game Theory and its Applications*. American Mathematical Society, 1981.

[27] N. Alexandrov. *Multilevel Algorithms for Nonlinear Equations and Equality Constrained Optimization*. PhD thesis, Rice University, 1993.

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

[29] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., 1983.

[30] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. Technical Report SOL 86-2, Dept. of Operations Research, Stanford University, Jan. 1986.