Collaborative Optimization: Status and Directions

Ilan Kroo*, Valerie Manning+
Stanford University
Stanford, CA 94305

* Professor, Dept. of Aeronautics and Astronautics, Associate Fellow AIAA
+ Post-Doctoral Student, Dept. of Aeronautics and Astronautics
Copyright ©2000 by Ilan Kroo. Published by the American Institute of Aeronautics and Astronautics, Inc., with Permission.
Abstract
This paper describes the current status of research on collaborative optimization, a strategy for
distributed design, and its application to large-scale aerospace systems. This approach to
computation-based design provides a method by which design tasks may be decomposed into
domain-specific subproblems, and coordinated to achieve an optimal system. Developed over
the past few years by researchers at Stanford, NASA, Boeing, and other universities, the
methodology is now being applied to large-scale problems requiring high-fidelity modeling. The
paper describes some of these applications, summarizes recent results, and suggests variants of
collaborative optimization that have been found to improve overall performance.

Introduction and Background
Most applications of multidisciplinary optimization initially involved the direct integration of
multiple disciplinary analyses and an optimizer. For small problems, wiring together such a
system is quite feasible and leads to a usually efficient, but often hard to explain, procedure. As
computational capabilities grew, engineers scaled this approach to larger problems and its
limitations became apparent. The need for analysis and data management became better
recognized and a second generation of MDO methods came into use. Distributed analysis
systems could utilize multiple computers, increasing the practical scale of MDO problems.
Database management and modular analysis coordination improved efficiency and maintainability.
Despite these improvements, the reliance on a central optimizer as decision maker on all matters,
is not a practical approach to large-scale system design. This deficiency has led to the
development of what may be considered the third generation of MDO methods: strategies for
distributed design optimization. These three types of MDO systems are illustrated in figure 1.

![Figure 1. Three generations of MDO architectures: integrated, distributed analysis, distributed design.](image)

In fact, distributed design is the way any large-scale system is designed now. However, it is
often the case that the specific approach to distributed design is not well-planned. Sequential
disciplinary designs and informal iteration can lead to designs that are sub-optimal. An
important current challenge for large scale MDO is the development of a practical method for
distributed design optimization that is, at least, not terribly inefficient and leads to good designs.
Several strategies have been proposed over the past twenty years including a variety of multi-level optimization methods [1-9]. This paper deals with one such approach that has found some success, partly because it closely matches the way complex system design is actually accomplished. Collaborative optimization is described in more detail in [10-18], but consists basically of a bi-level optimization architecture in which individual disciplinary teams are charged with satisfying local constraints. These subspace design teams are permitted to vary local parameters to accomplish that task. Since there may not exist sufficient local degrees of freedom to satisfy all of the constraints, subspaces are permitted to depart from the values of interdisciplinary parameters, established as targets by the system-level coordination method, although this departure is to be minimized. Thus, it is the job of the subspaces to satisfy constraints while working to define a design that everyone can agree on – hence the name collaborative optimization, or CO. The system is charged with adjusting the target values so that such agreement is possible while minimizing the system-level objective. The basic layout is shown in figure 2.

![Figure 2. Basic structure of a collaborative optimization problem.](image)

Figure 3 provides a graphical view of this process. The system chooses a set of targets, depicted by the point P. The subspaces then adjust their degrees of freedom in order to satisfy their own constraints while trying to match this target point as closely as possible. One may think of the subspaces as constrained to move along the lines shown, but connected to the target point with springs. The system tries to move the target in such a way that the subspace points coincide at the lowest possible objective value.

![Figure 3. Graphical view of CO process.](image)
Several choices are possible for the forms of the subspace objectives and the system-level compatibility constraints. These have been investigated in a number of previous studies and it seems likely that improved selections will be identified [18]. Many of the fundamental characteristics of CO are apparent in its various forms, and most of the applications we have studied relied on simple L2 measures for the subspace objectives (i.e. the sum of the squared differences between the local values for a variable and the requested system targets). When the subspace objectives are used directly as the system level constraints, simple post-optimality sensitivities can be used to efficiently guide the system optimizer to an improved design [19]. Note that the use of the sum-of-squares corresponds to a hardening spring in the model of figure 3, meaning that as the subspace solutions get closer, the tension falls quickly. This sometimes causes difficulties as discussed in the subsequent section.

In collaborative optimization, unlike some multi-level optimization approaches, we must assure that each subspace is given sufficient degrees of freedom that it can always achieve a design that is feasible with respect to its local constraints. This is done by making copies of the system target variables and allowing the subspace to vary these copies. Although this increases the dimensionality of the subproblem, it avoids difficulties that have been encountered in some multi-level optimization methods when responsibilities for design variable selection and constraint satisfaction are distributed among levels. As an example consider the following simple problem with only a single subspace. The goal is to minimize \( J = x^2 + y^2 \) with respect to \( x \) and \( y \), subject to the constraint that \( x = y + 2 \). Imagine that this is posed as a multilevel problem in which the subspace is asked to vary \( y \) to minimize \( J \) while the system varies \( x \) to satisfy the constraint and minimize \( J \) as well. In this case, although both the subspace and system are trying to minimize the same function, the subspace always chooses \( y = 0 \), since this minimizes \( J \) for any given value of \( x \), forcing the system to choose \( x = 2 \) and producing \( J = 4 \). The correct solution is clearly \( y = -1 \) and \( x = 1 \) with \( J = 2 \). This solution is found by collaborative optimization and most other schemes, but it illustrates that one must be cautious in dividing responsibilities for constraint satisfaction.

Details of the collaborative method have been developed by several researchers, primarily in the last ten years, although closely-related work is described in publications as early as 1977. Peterson et al. [20] describes a method in which subspaces are responsible for constraint satisfaction and CO-like targets are used. Thareja and Haftka [3] experimented with a very similar scheme with quadratic subspace objectives, encountering numerical difficulties, partly inherent in the formulation and partly associated with the optimizer. Sobieski, Riley, and Barthelemy [4,5] explored schemes in which the subspaces tried to satisfy a cumulative constraint. This is the alternative concept to collaborative optimization, in which interdisciplinary compatibility is maintained at each system-level iteration at the expense of disciplinary feasibility. It may be argued that this is an advantage, although in many cases a single infeasible design is no better than a set of not quite compatible designs. Related methods in which each discipline models constraints in other disciplines and is allocated some responsibility for constraint satisfaction are sometimes termed concurrent subspace optimization (CSSO) methods are described in references
such as [21] and [22]. More recent algorithmic development of CO and CO-like methods are described in [14-18].

Along with this method development, several researchers have applied CO to both simple test problems [10,18,23,24] and more complex design problems [25-28]. The latter have included applications to launch vehicle design, trajectory optimization, aeroelastic design, supersonic aircraft optimization, conceptual bridge design, undersea vehicles, and unmanned aircraft. This experience has highlighted both the advantages and problems associated with collaborative optimization; these are summarized in the following section.

CO Features and Issues

Several advantageous features of collaborative optimization have been demonstrated in these applications, although little quantitative information is available to demonstrate these points (see [14], however, for some data of this sort). And since CO is still relatively immature, little experience in true industrial environments is available. The examples cited here do show that the software integration task is expedited, since the process can be asynchronous and communication requirements are minimized. In addition, the computational effort can be easily divided among multiple platforms with heterogeneous hardware and software, including different optimizers for different disciplines. It is also anticipated that in an actual industry implementation the maintenance of disciplinary design variables and constraints within each disciplinary design task will be advantageous, matching closely the structure of existing practice. Many of the anticipated beneficial features of the architecture only exist for large, multidisciplinary problems, making it difficult to demonstrate with typically small test problems. Problems for which we expect CO to be best suited include the following:

- Large problems requiring significant effort for subproblem solution (integrated optimization or single-level schemes are probably better suited to small problems)
- Problems with low dimensionality coupling, requiring few system-level design variables.
- Problems that may exploit special methods for disciplinary optimization, such as sparse optimization in a trajectory subproblem, adjoint-based aerodynamic design, or linear methods.
- Situations in which the organizational environment makes tight integration undesirable.

Certain features of the architecture have created difficulties, especially for inexperienced developers. Our experiments have suggested that the price that must be paid for the advantages of decomposition is a somewhat increased computational time. However, some studies have cited extremely large computational increments, which may usually be attributed to implementation details. Several features of the method contribute to the ease by which one may arrive at a poor implementation. First, any multi-level distributed system is complex and with multiple optimizers and distributed databases, the initial implementation can be daunting. Secondly, the problem formulation and decomposition requires careful planning, a step that is not required for an integrated system. Finally, certain mathematical details can cause numerical difficulties for many existing optimization methods. In common with most multi-level schemes, the CO system-level problem may be sensitive to the selection of subspace optimization parameters such as feasibility or optimality tolerances. In some variants of CO, changes in the subspace active
constraint set can lead to non-smooth behavior of the system-level constraints. Although decomposition may not introduce spurious local minima itself, the design topology is different from the integrated problem and may be more difficult to interpret. We have already alluded to the implications inherent in the choice of system constraint form. This aspect of the method has received greater attention recently and warrants careful consideration as various versions of CO are developed.

The use of quadratic forms for the system level compatibility constraints means that near the solution, changes in system targets have little effect on the constraint values. Specifically, the gradient approaches zero, leading to difficulties for many optimizers, especially those that rely on linear approximations to these functions. This was observed in the early development of CO and led to slow convergence of the system near the solution as shown in figure 4 (from [13]).

![Figure 4. CO Convergence](image)

The implications of the singular Jacobian are discussed in [17] using an SQP method (NPSOL [29,30]) at the system level. For a simple test case, convergence of the system problem was not achieved. This failure is attributed, not to a failure of the optimizer, but to the specific CO formulation. In fact, the problem is related to the combination of the optimizer and formulation details, since the exact problem and formulation used in [17] can be solved in a relatively efficient and very robust manner by other numerical optimizers (even the solver in Microsoft Excel). This is not to say that this feature of the formulation is to be ignored. Indeed SQP methods are some of the most efficient methods available for constrained problems and most of the CO problems that we have studied have also employed NPSOL. In [13] a large set of quadratic test problems was studied using a version of CO with NPSOL, achieving convergence in every case, so the difficulties cited may require additional consideration, but improvements in this aspect of CO should certainly be pursued.

Some alternate choices for the form of subspace objectives, system constraints, and optimizers are discussed in [18] and appear promising. Two other approaches have also been studied in this regard: the use of a less efficient system optimizer that is more tolerant of non-smooth functions, and the use of response surfaces to model the system constraints. The latter avoids the problem with singular Jacobians, not by smoothing the functions it represents, but by providing such an inexpensive representation of the function that very robust but inefficient optimizers are perfectly acceptable. This approach is discussed in somewhat more detail in the following section and in Refs. [31, 32].
A problem similar to that described in [17], but which may be more representative of the type of problem well-suited to CO, is shown in figure 5. This problem is provided as a simple test problem for those interested in experimenting with the architecture and as an illustration of some features of approximation methods in a later section.

Minimize: $$J_{sys} = a_1^2(x_1, x_2) + 2a_2^2(x_2, x_3) + a_1 a_2$$

Subject to:

\[
\begin{align*}
    x_1 + x_2 &< 1; \\
    2a_1 + x_2 &< x_1; \\
    x_3 - x_2 &< -2; \\
    2a_2 + x_2 &= x_3
\end{align*}
\]

Solution: $$J_{sys} = 1.75$$

\[
\begin{align*}
x_1 &= 0, & x_2 &= 1, & x_3 &= -2, & a_1 &= a_0, & a_2 &= -1
\end{align*}
\]

Figure 5. Example problem

The problem may be decomposed into two subproblems as shown in figure 6. In this case, two of the three system targets affect each subproblem. Copies of these targets along with a single local variable comprise the set of subspace design variables. Note again that this is not the kind of problem that ought to be solved using CO, as it is simply (and much more efficiently) solved in the integrated manner of figure 5. If the subproblems each had hundreds of additional local variables and constraints this would be a more compelling, but less illustrative example.

\[
\begin{align*}
    \text{System} & \quad \text{Minimize: } J_{sys} = a_1^2 + 2a_2^2 + a_1 a_2 \\
                & \quad \text{with respect to: } a_1, a_2, x_2 \\
                & \quad \text{subject to: } J_1, J_2 = 0
\end{align*}
\]

\[
\begin{align*}
    \text{Subspaces} & \quad \text{Minimize: } J_1 = (a_1, a_1) + (x_2, x_2)^2 \\
                     & \quad \text{with respect to: } a_1, x_1, x_2 \\
                     & \quad \text{subject to: } x_1 + x_2 < 1 \quad 2a_1 + x_2 = x_1
\end{align*}
\]

\[
\begin{align*}
    \text{Subspaces} & \quad \text{Minimize: } J_2 = (a_2, a_2) + (x_2, x_2)^2 \\
                     & \quad \text{with respect to: } a_2, x_2, x_3 \\
                     & \quad \text{subject to: } x_3 - x_2 < -2 \quad 2a_2 + x_2 = x_3
\end{align*}
\]

Fig. 6. CO version of example problem

The solution to the first subspace optimization problem is shown in figure 7. The optimal subspace objective, $$J_1^*$$, varies quadratically with respect to the system targets, $$a_1$$ and $$x_2$$ on the right of the $$J_1^* = 0$$ line, and is zero over the entire lower left region of the space.
Despite the least squares form of the subspace objectives, the system converges rapidly using a number of optimizers.

**CO in Practice**

Since many of the advantages of CO and problems with integrated design were expected only in larger scale design problems, we undertook a more ambitious test problem. The idea was to study a sample application that employed distributed, high-fidelity modeling, more representative of a problem that might be solved by industry using CO. The supersonic aircraft design problem included three major disciplines shown in figure 8. The analysis codes were not developed specifically for this problem and included a Boeing aerodynamics code, a finite element code obtained from NASA Langley, and a mission analysis that included a look-up table for engine data.

Two supersonic aircraft concepts were studied: a conventional design based on a reference Boeing HSCT, and a natural laminar flow concept. Details are available in [28] but are summarized here. Several problems were investigated, but generally involved 12-16 system design variables and many design variables local to the disciplines. The goal was to minimize take-off weight subject to typical mission performance, aerodynamics, and structural constraints.

Aerodynamic analysis was required primarily for loads and cruise performance. This was accomplished using the Boeing code A502, a higher-order panel method suitable for subsonic and
Results from this code were compared with Euler solutions at the conditions of interest and found to be suitable, given the goals of this project. This selection allowed us to employ an automated surface paneling program rather than developing a grid generation tool for this problem. Since the A502 code solves the inviscid flow equations, viscous skin friction and viscous-related pressure drag was added using conventional advanced design methods. Solutions for loads and drag were obtained at an initial cruise point and at a single structural design point. These were used — or rather the corresponding system targets were used — by the mission and structures disciplines as inputs for the corresponding analyses. Although the use of only two design points is hardly realistic, it provided an opportunity to consider how multiple flight conditions might be handled in this framework. And despite the limited number of design conditions, the handling of thousands of shared pressures and deflections among the aerodynamics and structures disciplines to achieve an aeroelastic solution provided sufficient challenge here (see [28]).

In the laminar flow case, a transition model based on the method of Ref. [33] was also included.

Structural analysis was accomplished using the code FESMDO, a finite element analysis method, developed for NASA Langley. This program, which was executed remotely at Langley, was used to obtain stresses and deflections of the wing. The results from this program were first compared with those from NASTRAN for representative models of interest in this problem. The models themselves consisted of elements depicted in figure 9 and were created by a parametrically-driven model generator. Non-optimum factors and statistical weights for other aircraft components were derived based on previously published NASA and McDonnell Douglas studies.

![Finite Element Model](image)

**Figure 9. Wing structural model.**

A mission analysis discipline was used to compute take-off and landing field lengths, range, climb gradient, and available fuel volume. This “discipline” also included an engine model derived from HSCT studies.

The solution of the system was accomplished using an approximation method described in the
next section. Approximately a dozen update cycles were required to obtain “convergence” of the system problem, which in this case was formulated using a penalty function method. Figure 10 shows the decreasing take-off gross weight and the objective:

\[ J = TOGW + k (J_{aero} + J_{structures} + J_{mission}) \]

![Figure 10. Reduction of weight and augmented objective with design cycles.](image)

Subsequent cycles (to about 20) showed little change in the system target values. The results of this application experiment were encouraging but suggested that additional work is required before such an undertaking would be practical in a project environment. Some of these issues and possible approaches are described in the following section.

**Directions**

Improved usability and efficiency of the CO process for large problems will be enhanced by developments in two general areas: the optimization problem itself, and the software implementation. A number of variations on the CO theme are currently under investigation. In addition to subtle, but important, modifications to the formulation such as those in Ref. [18], more significant changes may be beneficial.

**Approximate Modeling and CO**

One possibly critical element of a practical CO method is the efficient incorporation of approximation models. This has been addressed initially in [31], but additional developments are likely. Multi-level methods may be better suited to, and benefit more from, the use of approximate modeling than other optimization methods. Techniques such as response surfaces are particularly appealing in CO for several reasons. When used to model the results of subspace optimization, the dimensionality of the response surface can be much smaller than would be required for fitting an integrated analysis system. Problems with low dimensionality interdisciplinary coupling, natural candidates for solution by CO, also gain most using this
approach. Along with the usual response surface features that aid in parallel execution and load balancing, two other considerations suggest the important synergy that exists in this combination. As has been mentioned, the use of approximate models for the system constraints sidesteps an important efficiency difficulty with some versions of the method. In this way response surfaces can help CO. Conversely, we can exploit knowledge about the problem itself to improve the efficiency of the approximate model generation.

Efficient model generation for CO is enabled by the fact that we do not require a method that approximates a general function, only functions that are solutions to CO subproblems. As an example, consider the simple test problem defined previously. We wish to fit the optimum subspace objective, $J_1^*$, shown in figure 7, as a function of the system targets $a_1$ and $x_2$. As can be seen from the figure, the function is quadratic in each region of the solution, but cannot be fit globally with a single quadratic approximation. However, even if all we desired was a local quadratic fit, we would have to solve the subproblem at least six times. The solution to the subproblem provides much more than the value of $J_1^*$, though, and this information can be used to great advantage. Assume that, as shown in figure 11, the first target values are given as $a_1=2.5$, $x_2=0$. The subspace solution in this case is $a_1'=1.5$, $x_2'=-1.0$. Since the subspace returns these values, we are assured that if we were to specify $a_1=1.5$, $x_2=-1.0$, the subspace could satisfy its constraints and match these targets exactly. So although we have evaluated only one point in the 2D space, we know the solution at two points. A quadratic model could therefore be built with as few as three subspace evaluations. Moreover, not only do we know that the $a_1=1.5$, $x_2=-1.0$ target values lie on the $J_1^*$=0 contour, we also know that this is the closest point on the contour to our initial point, so the contour must be normal to the vector connecting these two points as shown in figure 11. Combined with other knowledge about the formulation, we can obtain an exact fit in this case with only one subspace optimization. Of course the simple test problem was quadratic and for general problems we would not know this, so it would generally take several approximation refinement steps (e.g. using trust region methods such as those in [34,35] -- these steps are the design "cycles" shown in figure 10) before a reasonable approximation was achieved; nonetheless, this represents a large improvement in efficiency.

Fig. 11. Function knowledge used to construct fit
Related Problems in AI and Control

The basic concept of collaborative optimization and distributed design is very closely related to topics of interest to the AI community and to distributed control problems. Continuing advances in these fields may be significant in the continued development of CO.

Distributed MDO may be viewed as a distributed control problem. Researchers in control theory are concerned with the control of multiple cooperating robots and even optimal multi-agent collaboration. Although the tasks associated with individual robots are simple compared with the tasks of disciplinary design teams, the coordination problem is quite similar. New techniques are prevalent in this field.

Researchers in AI are concerned with self-organizing systems in which individuals are given certain local objectives and in achieving these produce an emergent system behavior, possibly a desirable, or even optimal system behavior. This is, of course, a perfect description of CO. Techniques for creating systems with desired emergent behavior are being developed (e.g. Ref. [36]). These may be of interest in the CO meta-design problem: choosing subspace objectives to assure good system behavior.

Implementation

One of the most significant impediments to the development and application of collaborative optimization (and other distributed design approaches) is the work required to implement and deploy the associated infrastructure. These systems may involve multiple levels of optimization, different optimizers, response surface methods, trust-region model management, communication protocols (sometimes secure communication), and even engineering analyses. The complexity of such an enterprise system makes the implementation of a real system for CO a major undertaking. Even students wishing to apply CO to an example problem must develop a significant infrastructure that can become a research project in its own right. This suggests the need for a framework that facilitates the use of CO or related methods.

Existing frameworks are often poorly suited for multilevel distributed design. Although tools such as iSight, ModelCenter, WindChill, and others provide optimization and visualization capabilities, it is not an easy task to incorporate a custom optimization routine into a subproblem, for example, and many of these systems require substantial investments in training, licensing, and consulting. New software tools are becoming available that make the development of complex enterprise systems more manageable and provide an opportunity for an easy-to-use CO-friendly distributed design environment. Such a framework is currently being developed and tested.

Caffe, a Collaborative Application Framework For Engineering, is a general framework that supports distributed analysis and design and is well-suited for collaborative optimization. Leveraging commercial software tools such as Sun Microsystems’ Java 2 Enterprise Edition and emerging XML software, the system is standards-based, with secure network communication, and provides support for legacy applications.

The system simplifies the integration of existing code (analyses, RSM, optimizers) for project
developers, manages the design process on multiple distributed platforms, and provides an easily customizable, browser-based user interface. Caffe is designed from the start for large scale distributed design and therefore features multi-threaded, asynchronous execution of subtasks, allowing remote connections to fail and restart without stopping the overall design process.

Simple or complex design structures are built from components, that may be geographically distributed and include a network-aware application, an associated XML database, and a graphical user interface (Figure 12).

![Component Diagram](image1.png)

**Fig. 12.** Caffe component includes application, data, and user interface elements.

Applications may be analyses, simple wrappers for legacy codes, or design tools such as optimizers or response surface generators. Figures 13 and 14 show how collaborative optimization is structured using components in the Caffe system. Figure 14 shows the structure of the method used in the supersonic aircraft design problem described previously.

![System Diagram](image2.png)

**Figure 13.** CO structure in Caffe.
The framework relies heavily on XML and XSLT technologies for distributed data storage and to simplify user interaction with the system. Figure 15 shows how users view the same data in different ways using a browser that supports XML, XSL, and applets or with custom editing tools.

**Fig. 15. Various views of the same XML data in Caffe.**

**Conclusions**

Collaborative optimization, while still maturing, has been used successfully in many applications. It is ideal for integrating *design* codes that include their own optimizer as is often the case for trajectory optimization or optimal control design and for large scale problems with low dimensionality coupling between the subproblems.

The approach can be inefficient when not carefully employed. As with any optimization problem, the user must match the optimization algorithm to the problem of interest and understand issues related to scaling and function smoothness. Work remains on improving the robustness and efficiency of the system-level coordination problem through variations in the form of the subspace objectives and system constraints or through optimization algorithms better suited to the system-level problem.

CO with response surfaces that model subspace design results continues to appear promising. New general fitting techniques such as kriging may be useful here, but custom methods tailored to subspace fitting may be more efficient. New approaches to reduced basis modeling and trust
A software framework for easy CO implementation is needed to reduce the effort required for deployment and to improve robustness. A prototype system is under development and shows potential for making distributed design optimization more accessible.

Acknowledgements

Much of the work described here has been accomplished by students in the Aircraft Aerodynamics and Design Group at Stanford (Bobby Braun, Ian Sobieski, Peter Gage, and others). We’d also like to gratefully acknowledge support from NASA’s Langley and Ames Research Centers as well as from the Boeing Company and the always stimulating interactions with our colleagues there.

References


