AIAA JOURNAL
Vol. 35, No. 7, July 1997

Sequentially Decomposed Programming

Sigurd A. Nelson II* and Panos Y. Papalambros†
University of Michigan, Ann Arbor, Michigan 48109

Model-based decomposition is a powerful tool for breaking design problems into smaller subproblems, establishing hierarchical structure, and analyzing the interrelations in engineering design problems. However, the theoretical foundation for solving decomposed nonlinear optimization problems requires further work. We show that the formulation of the coordination problem is critical in quickly identifying the correct active constraints and that solving subproblems independently may hinder the local convergence of algorithms tailored to hierarchical coordination. Yet hierarchical decomposition algorithms can have excellent global convergence properties and can be expected to exhibit superior improvement in the first few iterations when compared to the undecomposed case. Based on these insights, a generic sequentially decomposed programming (SDP) algorithm is outlined. SDP has two phases: far from the solution (first phase) decomposition is used; close to the solution (second phase) subproblems are not solved separately. The generic SDP is applied to sequential quadratic programming (SQP) to define an SDP–SQP implementation. A global convergence proof and a simple example are given.

Nomenclature

$\boldsymbol{B}$	= approximate Hessian
$\boldsymbol{B}_j$	= approximate Hessian for the jth subproblem
$\boldsymbol{B}_{jjj}$	= elements corresponding to $\boldsymbol{x}_j$ for the approximate Hessian for the jth subproblem
$\boldsymbol{d}$	= search direction
f	= system level objective function
f_j	= jth term in the system level objective function, corresponds to objective function for the jth subproblem
$\boldsymbol{g}$, $\boldsymbol{h}$	= vector of inequality/equality constraints
$\boldsymbol{g}_j$, $\boldsymbol{h}_j$	= vector of inequality/equality constraints associated with the jth subproblem
g_{ji}, h_{ji}	= scalar components of the vector of inequality/equality constraints associated with the jth subproblem
$\boldsymbol{H}$	= Hessian
$\boldsymbol{H}_j$	= Hessian for the jth subproblem
$\boldsymbol{H}_{jjj}$	= elements corresponding to $\boldsymbol{x}_j$ for the Hessian for the jth subproblem
$L(\boldsymbol{x}, \lambda, \mu)$	= Lagrangian function
$L_j(\boldsymbol{x}_0, \boldsymbol{x}_j, \lambda_j, \mu_j)$	= jth term in the Lagrangian or the Lagrangian function of the jth subproblem
m_{eq}, m_{ineq}	= number of equality/inequality constraints
$m_{j,\text{eq}}$, $m_{j,\text{ineq}}$	= number of equality/inequality constraints in the jth subproblem
n	= number of variables
n_j	= number of variables in the jth subproblem
$\boldsymbol{x}$	= vector of design variables
$\boldsymbol{x}_j$	= vector of design variables in the jth subproblem
$\boldsymbol{x}_0$	= vector of linking variables
α	= step length
λ, μ	= vector of multipliers associated with the inequality/equality constraints
$\lambda_j \mu_j$	= vector of multipliers associated with the inequality/equality constraints of the jth subproblem
$\lambda_{ji} \mu_{ji}$	= ith component of the vector of multipliers associated with the inequality/equality constraints of the jth subproblem
$\partial \boldsymbol{x}_j^{\dagger}(\boldsymbol{x}_0)/\partial \boldsymbol{x}_0$	= infinitesimal change in the variables of the jth subproblem with respect to the linking variables
∇	= gradient operator
$\nabla_{\boldsymbol{x}_0}$	= gradient operator with respect to the linking variables

Subscripts

i	= indices, scalar components of the constraints
j	= indices the subproblems

Superscripts

k	= iteration counter
†	= quantity after solving the subproblem

Received Aug. 22, 1996; revision received April 22, 1997; accepted for publication April 2, 1997.

*Ph.D. Candidate, Design Laboratory, Department of Mechanical Engineering and Applied Mechanics.

†Professor, Design Laboratory, Department of Mechanical Engineering and Applied Mechanics. Member AIAA.

I. Introduction

A. Overview of Work and Related Research

OPTIMIZATION methods have been applied with practical success to individual components of a system using well-developed models and simulations. At the component level, the physics, design goals, and other modeling issues are such that computer automation is a relatively straightforward task: simulations are developed typically by the same people with similar interests and, therefore, often have a consistency allowing the simulations to work well together. Difficulties arise when design must be performed at the system level, a system being a collection of connected components or processes. Different computational models are coupled via common design quantities, and the utility of the design must reflect how the system performs as a whole.

If an optimal design problem is stated as a nonlinear program (NLP):

Minimize $f(\boldsymbol{x})$
subject to

$$\begin{aligned} g_i(\boldsymbol{x}) &\leq 0 \qquad i = 1, \ldots, m_{\text{ineq}} \\ h_i(\boldsymbol{x}) &= 0 \qquad i = 1, \ldots, m_{\text{eq}} \end{aligned} \tag{1}$$

where n, m_{ineq}, and m_{eq} are large and f, g_i, and h_i are nonlinear or computationally expensive to evaluate, then finding a solution $\boldsymbol{x}*$ is inherently difficult. Many researchers are currently of the opinion that the inherent structure of the problem must be exploited to reliably solve large NLP, for example, Conn et al.[1] and Papalambros.[2]

To exploit the structure of a design problem, the concept of structure must first be defined. Several such concepts exist. For example,

Conn et al.,[1] Lootsma,[3] Dantzig and Wolfe,[4] Azarm and Li,[5,6] and Sobieszczanski-Sobieski et al.[7] all work with different concepts of structure, each leading to a suitable optimization method. It is therefore necessary to use the concept of structure that most adequately fits the system description.

This article presents results of an ongoing investigation into the use of a particular type of structure (presented in Sec. I.B) coupled with a particular class of algorithms (presented in Sec. II.C).

B. Structure of NLP

The work presented here assumes structure to be based on optimal model-based partitioning as conceived by Wagner and Papalambros[8] and refined by Michelena and Papalambros.[9] When applied to large-scale NLP [Eq. (1)], functions are assigned vertices and variables are assigned hyperedges in a hypergraph. A hyperedge is an entity that connects two or more vertices. Thus, if two or more functions are dependent on the same variable, the corresponding vertices are connected with a hyperedge. A hypergraph is said to be disjoint when at least one vertex cannot be reached from at least one other vertex by following a path of hyperedges. A hypergraph is optimally partitioned when a predetermined number of properly sized, disjoint hypergraphs remain after the removal of a minimal number of hyperlinks.

If a suitable partition for an NLP is found, the functions and variables in Eq. (1) can be reordered and grouped as follows.

Minimize $\boldsymbol{x} \in \mathcal{R}^n$:

$$f_0(\boldsymbol{x}_0) + \sum_{j=1}^{p} f_j(\boldsymbol{x}_0, \boldsymbol{x}_j) \tag{2}$$

subject to

$$\begin{aligned} g_{0i_0}(\boldsymbol{x}_0) &\leq 0 \\ h_{0i_0}(\boldsymbol{x}_0) &= 0 \\ g_{ji_j}(\boldsymbol{x}_0, \boldsymbol{x}_j) &\leq 0 \\ h_{ji_j}(\boldsymbol{x}_0, \boldsymbol{x}_j) &= 0 \end{aligned} \qquad \begin{aligned} &j = 1, \ldots, p \\ &i_j = 1, \ldots, m_{j,\text{ineq}} \\ &i_j = 1, \ldots, m_{j,\text{eq}} \end{aligned}$$

The vector of variables $\boldsymbol{x}_0 \in \mathcal{R}^{n_0}$ common to most functions corresponds to the removed hyperlinks and is termed the vector of linking variables. If the linking variables are held constant, then Eq. (2) can be rewritten as the sum of p different independent subproblems, as follows.

Minimize $\boldsymbol{x}_j \in \mathcal{R}^{n_j}$:

$$f_j(\boldsymbol{x}_0, \boldsymbol{x}_j) \tag{3}$$

subject to

$$\begin{aligned} g_{ji_j}(\boldsymbol{x}_0, \boldsymbol{x}_j) &\leq 0 \qquad i_j = 1, \ldots, m_{j,\text{ineq}} \\ h_{ji_j}(\boldsymbol{x}_0, \boldsymbol{x}_j) &= 0 \qquad i_j = 1, \ldots, m_{j,\text{eq}} \end{aligned}$$

As an example, consider a problem that first appeared in Ref. 10 and was used as a decomposition example in Ref. 11.

Minimize $R, L, t_s, t_h \geq 0.1$:

$$f_1 + f_2 + f_3 + f_4 = 0.662RLt_s + 1.777R^2t_h + 1.58RLt_s^2 + 19.48Rt_s^2 \tag{4}$$

subject to

$$\begin{aligned} g_1 &= 0.131R - t_h \leq 0 \\ g_2 &= 0.0193R - t_s \leq 0 \\ g_3 &= 413000 - R^2L \leq 0 \\ g_4 &= 0.00417L - 1.0 \leq 0 \end{aligned}$$

Figure 1 displays the NLP in Eq. (4) as a hypergraph. The design quantities $\{R, L, t_h, t_s\}$ are drawn as amorphous shaded objects (the hyperlinks), which link every function that depends on the particular design variable. The hyperlink L connects the vertices $\{f_1, f_3, g_3,$ and $g_4\}$ because $\{f_1, f_3, g_3,$ and $g_4\}$ are all functions of L, and the hyperlink R connects the vertices $\{f_1, f_2, g_1, g_2,$ and $g_3\}$ because $\{f_1, f_2, g_1, g_2,$ and $g_3\}$ are all functions of R.

Decomposing the NLP is analogous to removing hyperlinks and finding disjoint subgraphs; namely, there is no path between certain sets of vertices. In Fig. 1, the hyperlinks corresponding to R and L have been removed, so that no paths exist between the subgraphs corresponding to subproblem 1 and subproblem 2. Thus, the two subsystems can be stated as two smaller NLPs. (Note that the objective functions have been renumbered to fit the numbering scheme for the decomposition.)

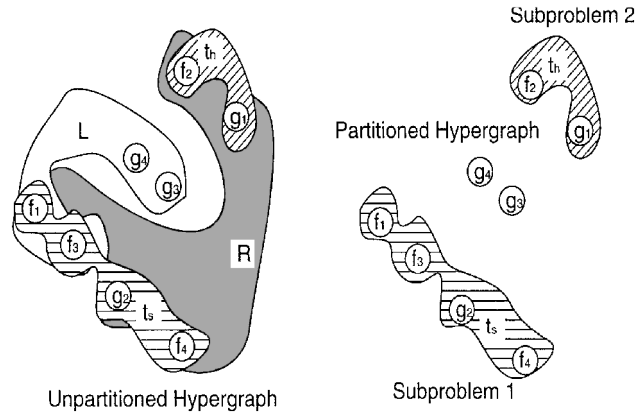

Fig. 1 Hypergraph representation of Eq. (4) (left) before and (right) after partitioning.

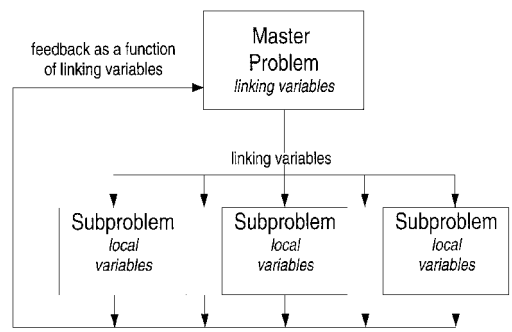

Fig. 2 Top down authoritative structure of a hierarchical coordination strategy.

Minimize $t_s \geq 0.1$:

$$f_1 = 0.662RLt_s + 1.58Lt_s^2 + 19.84Rt_s^2 \tag{5}$$

subject to

$$g_{1,1} = 0.0193R - t_s \leq 0$$

Minimize $t_h \geq 0.1$:

$$f_2 = 1.777R^2t_h \tag{6}$$

subject to

$$g_{2,1} = 0.131R - t_h \leq 0$$

C. Structure of Algorithms

The appeal of the structure defined in Sec. I.B is that each subproblem is smaller (in dimension, etc.) and, therefore, easier to solve, provided that the optimal value of the linking variables $\boldsymbol{x}_0^*$ is known. In fact, this structure naturally leads to a hierarchical approach to solving the original NLP. Such a hierarchical approach is outlined by the following steps.

1) Subproblems: holding the linking variables constant at the value $\boldsymbol{x}_0 = \boldsymbol{x}_0^k$, for each subproblem $j = 1, \ldots, p$, find the minimizer of the jth subproblem, $\boldsymbol{x}_j^\dagger(\boldsymbol{x}_0)$ [the solution to Eq. (3)].

2) Coordination: find a value for $\boldsymbol{x}_0^{k+1}$ that will improve the (overall) objective and/or maintain feasibility.

3) Repeat until some convergence criteria are met.

Hierarchical coordination algorithms are often portrayed as a top-down authority structure (Fig. 2). Many researchers have suggested approaches that fit within the preceding for engineering design problems.[7,12,13]

However, Vanderplaats and Yoshida[14] have shown that $\partial \boldsymbol{x}_j^\dagger(\boldsymbol{x}_0)/\partial \boldsymbol{x}_0$ need not be a continuous function, which can create difficulties in solving the coordination problem. The discontinuities in $\partial \boldsymbol{x}_j^\dagger(\boldsymbol{x}_0)/\partial \boldsymbol{x}_0$ occur because of a change in the active set of constraints, and Vanderplaats et al.[13] reformulated the coordination problem to include all variables and constraints in a sequential linear programming approach in order to overcome this difficulty. One conclusion of Vanderplaats et al.[13] was that decomposition will probably not lead to advances in computational efficiency unless additional measures such as distributed computing are used. The work here is specifically addressing convergence and efficiency issues.

In Sec. II, two additional theoretical concerns with coordination of hierarchical decomposition problems are presented. In Sec. III, sequentially decomposed programming (SDP) is outlined as a generic method to overcome the difficulties presented in Sec. II, with a specific application to sequential quadratic programming (SQP). In Sec. IV, a proof of global convergence is given. Section V gives an example of the performance of SDP, Sec. VI discusses open issues, and Sec. VII the current state of ongoing work and conclusions.

II. Difficulties with Decomposition: Hierarchical Coordination Strategies

Within the work presented here, the focus is on computational methods. If superscript k represents the iteration counter, then $\boldsymbol{x}^{k+1}$ depends on the values of $f(\boldsymbol{x}^k)$, $g_i(\boldsymbol{x}^k)$, $i = 1, \ldots, m_{\text{ineq}}$, $h_i(\boldsymbol{x}^k)$, $i = 1, \ldots, m_{\text{eq}}$, and their respective gradients. It is important to remember that coordination methods for hierarchical strategies deal only with the direct consequences of changing the linking variables $\boldsymbol{x}_0$. In other words, if $\boldsymbol{x}_j^\dagger$ is the solution to the jth subproblem, it is necessary to estimate how the subproblems will behave, or how $\boldsymbol{x}_j^\dagger$, f_j, g_{ji}, and h_{ji}, will change with respect to $\boldsymbol{x}_0$. For infinitesimal changes in the linking variable, the rates of change of $\boldsymbol{x}_j^\dagger$, f_j, g_{ji}, and h_{ji} are known as sensitivities. Sensitivities can be discontinuous when the active set of constraints changes,[14] but the following two subsections present other possible problems with hierarchically decomposed methods. The first argument focuses on the effectiveness of hierarchical algorithms with respect to global convergence issues, and the second argument focuses on local convergence issues.

A. Difficulties in Determining the Correct Active Constraints

Consider the linearly constrained nonlinear program in Eq. (7).

Minimize $\boldsymbol{x} \in \Re^2$:

$$5x_0^2 - 2x_1x_0 + 4x_1^2 - 16x_0 - 12x_1 + 0.1e^{x_0-2} + 0.2e^{x_1-1} \tag{7}$$

subject to

$$g_1 = x_1 - 3 \leq 0$$
$$g_2 = -3x_0/4 + x_1 - 1 \leq 0$$
$$g_3 = x_0 - 4x_1 - 2.5 \leq 0$$

Equation (7) is hierarchically decomposed into a master problem [Eq. (8)] and a single subproblem [Eq. (9)].

Minimize $\boldsymbol{x}_0 \in \Re$

$$5x_0^2 - 16x_0 + 0.1e^{x_0-2} + f_1(\boldsymbol{x}_0) \tag{8}$$

Minimize $\boldsymbol{x}_1 \in \Re$

$$f_1 = -2x_1x_0 + 4x_1^2 - 12x_1 + 0.2e^{x_1-1} \tag{9}$$

subject to

$$g_{1,1} = x_1 - 3 \leq 0$$
$$g_{1,2} = (-3x_0/4) + x_1 - 1 \leq 0$$
$$g_{1,3} = x_0 - 4x_1 - 2.5 \leq 0$$

At the point $\boldsymbol{x}^1 = [0, 1]$ (the linking variable $x_0 = 0$), we would like to know how f_1 changes as the linking variable x_0^1 is changed. The constraint $g_{1,2}$ is active at $\boldsymbol{x}^1$, and so it can be removed via direct algebraic substitution:

$$\left.\frac{\mathrm{d}x_1}{\mathrm{d}x_0}\right|_{g_{1,2}=0} = \frac{3}{4}$$

$$\left.\frac{\mathrm{d}f_1}{\mathrm{d}x_0}\right|_{\substack{g_{1,2}=0 \\ x_0=0}} = \begin{bmatrix} 11.5x_0 - 21 + 0.1e^{x_0-2} \\ -0.15e^{0.75x_0} \end{bmatrix} = -20.86 \tag{10}$$

Thus, at $\boldsymbol{x}^1$ it is possible to decrease the value of f_1 by increasing the value of x_0. A hierarchical coordination scheme using sensitivities would be similar to the algorithm that follows.

1) For $\boldsymbol{x}_0^k$, find the minimizer $\boldsymbol{x}_j^{\dagger k}$ for each subproblem $j = 1, \ldots, p$.

2) At $\boldsymbol{x}^{\dagger k} = [x_0^k \; x_1^{\dagger k} \; \ldots x_p^{\dagger k}]$, calculate $\mathrm{d}f_j^\dagger(\boldsymbol{x}_0^k)/\mathrm{d}\boldsymbol{x}_0$.

3) Set the value of $\boldsymbol{x}_0^{k+1}$ (changing $\boldsymbol{x}_j^{k+1}$, $j = 1, \ldots, p$ as predicted through the use of sensitivities) within acceptable move limits to decrease the value of the objective function $f = f_0 + \sum f_j$ and/or maintain/attain feasibility.

4) Repeat until some convergence criteria are met.

Figure 3 displays the contours of the objective function and the lines along which the constraints are satisfied for the NLP in Eq. (7). Starting from x^k, the next point produced by the algorithm must be along the ray (outlined in heavy black), which coincides with constraint $g_{1,2}$. Depending on the type of minimization used, the next

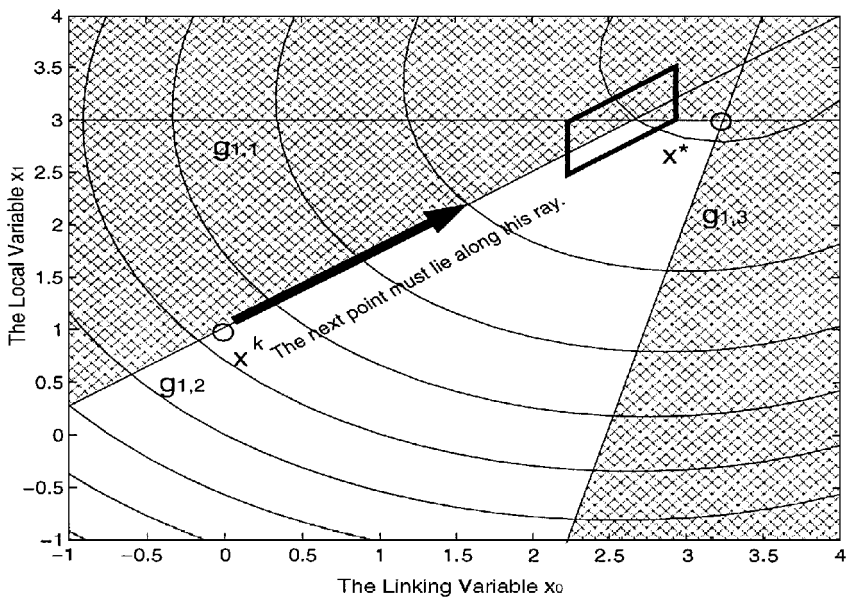

Fig. 3 Contour lines and constraints of the NLP stated in Eq. (7).

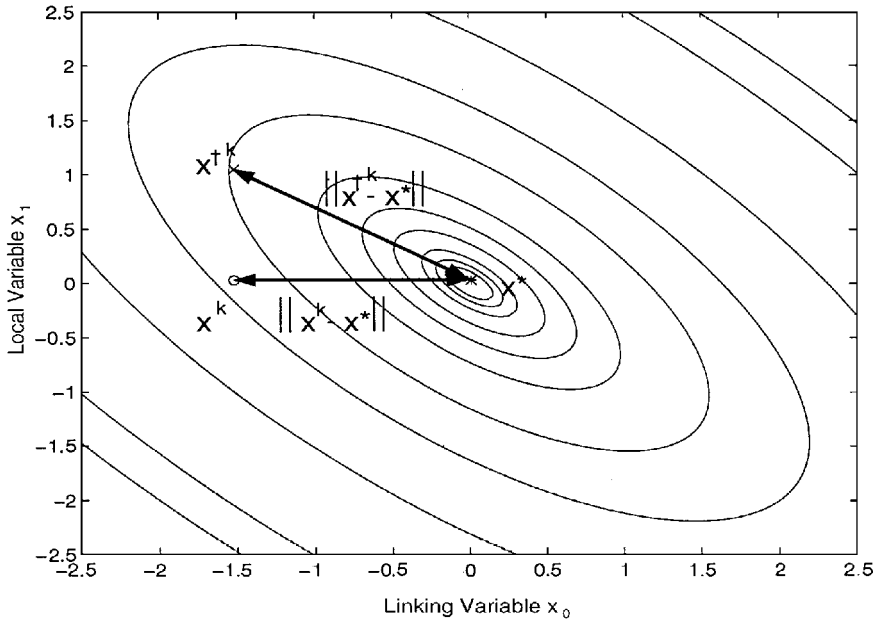

Fig. 4 Contour lines for the quadratic objective function Eq. (11).

point would probably be somewhere in the region marked as a parallelepiped. However, because of the definition of sensitivities and the way sensitivities are incorporated into the coordination problem, the point (in this case, the solution $\boldsymbol{x}*$ to the NLP) that identifies the correct set of active constraints does not lie along constraint $g_{1,2}$.

Put in more formal terms, suppose that the active inequality constraints at the solution are indexed $i = 1, \ldots, m^*_{\text{ineq}}$. Given a point $\boldsymbol{x}^k$ (not the solution), let χ^k represent the set of points which satisfy all equality and inequality constraints with indices $i = 1, \ldots, m^*_{\text{ineq}}$ when linearized at the point $\boldsymbol{x}^k$. Let X^k represent the set of points that can be reached using a particular coordination strategy, so that X^k is the feasible space of the coordination problem. If coordination is formulated as a decision making process with dimension n_0, then X^k is an affine subspace with dimension n_0 that includes the current point $\boldsymbol{x}^k$.

In Fig. 3, $\chi^1 \cap X^1 = \emptyset$, and so it is not possible to correctly identify the active set of constraints without extra iterations. Thus, if $\chi^k \neq \emptyset$, then the coordination problem should be formulated such that $\chi^k \cap X^k \neq \emptyset$. (Note that this does not necessarily require that the solution to the coordination problem identify the active set of constraints, only that the possibility should exist.)

It is possible to define sensitivities in terms of second-order derivatives, but the underlying problem illustrated here still stands: if coordination is formulated as a decision making process with dimension n_0, then there are instances in which the coordination problem cannot correctly identify the active set of constraints without extra iterations.

This is important because identifying the correct active set of constraints is both a condition eventually achieved by most algorithms and usually one of the sufficient conditions for whatever local convergence behavior is appropriate (see, for example, Refs. 15–17). In this simple example, coordinating with any approximate problem that used all of the variables and all of the constraints would have correctly identified the active constraints, whereas the coordination problem based on sensitivities could not. There are instances when using a larger dimensional problem for coordination is prohibitive, but one must consider the number of iterations in a large NLP, especially if function calls are expensive relative to solving the coordination problem. A coordination method that takes more computing power to solve may be beneficial if fewer iterations are needed.

B. Using Decomposition in the Neighborhood of the Solution

In this section, it is assumed that the active constraint set has been correctly determined and that the current iterate $\boldsymbol{x}^k$ is in the neighborhood of the solution $\boldsymbol{x}*$. By neighborhood, it is meant that if some traditional optimization algorithm were used, then the observed local convergence rate would be in accord with the theoretical analysis. When iterations are close to the solution, the focus of analysis is on local convergence, and so the distance between the current iteration $\boldsymbol{x}^k$ and the solution $\boldsymbol{x}*$ as measured by the Euclidean norm $\|\boldsymbol{x}^k - \boldsymbol{x}*\|$ is of particular interest. It is also assumed that whatever hierarchical algorithm is used, the algorithm fits the outline of model algorithm 1, so that given a starting point $\boldsymbol{x}^1$ the sequence of points produced is $\{\boldsymbol{x}^1, \boldsymbol{x}^{\dagger 1}, \boldsymbol{x}^2, \boldsymbol{x}^{\dagger 2}, \boldsymbol{x}^3, \boldsymbol{x}^{\dagger 3}, \ldots\}$.

Consider Eq. (11), an unconstrained quadratic objective function that has been decomposed into a master problem [Eq. (12)] and a single subproblem [Eq. (13)]:

$$\min_{\boldsymbol{x} \in \mathcal{R}^2} f = 1.5x_0^2 + 2x_0x_1 + 1.5x_1^2 \tag{11}$$

$$\min_{x_0} f = 1.5x_0^2 + f_1(x_0) \tag{12}$$

$$\min_{x_1} f_1 = 2x_0x_1 + 1.5x_1^2 \tag{13}$$

Figure 4 is a picture of the contour lines of Eq. (11). However, these could also be the contour lines of any convex function as seen in the plane uniquely determined by the solution $\boldsymbol{x}*$, the current iterate $\boldsymbol{x}^k$, and the iterate after solving all of the subproblems $\boldsymbol{x}^{\dagger k}$.

Starting from $\boldsymbol{x}^k = [-1.5, 0]$, we see that $\boldsymbol{x}^{\dagger k} = [-1.5, 1]$ is farther from the solution as measured by the norm $\|\boldsymbol{x} - \boldsymbol{x}*\|$, despite the fact that $f(\boldsymbol{x}^k) > f(\boldsymbol{x}^{\dagger k})$. Figure 4 highlights the fact that solving the subproblems while keeping linking variables constant may actually increase the distance from the current iterate to the solution, even though the objective function has decreased. In general, this nonintuitive fact makes the construction of an algorithm that guarantees $\|\boldsymbol{x}^k - \boldsymbol{x}*\| > \|\boldsymbol{x}^{k+1} - \boldsymbol{x}*\|$ theoretically very difficult. Also, an acceptable convergence rate may be impossible to prove. Regardless of the algorithm, if the purpose of optimization is to find the solution $\boldsymbol{x}*$, then it would be more prudent to devote computational power to reduce $\|\boldsymbol{x} - \boldsymbol{x}*\|$, as opposed to $f(\boldsymbol{x})$.

Both situations presented in Secs. II.A and II.B are problems with solving a nonlinear program using only a specific subset of variables at a time.

III. Sequentially Decomposed Programming

Two theoretical difficulties with traditional hierarchical decomposed methods were presented: 1) expressing $\boldsymbol{x}^{\dagger}_j(\boldsymbol{x}_0)$ can make it difficult to identify the correct set of active constraints and 2) in the neighborhood of the solution, hierarchical decomposition methods may have problematic convergence rates; furthermore, as discussed earlier, $\partial \boldsymbol{x}^{\dagger}_j(\boldsymbol{x}_0)/\partial \boldsymbol{x}_0$ is not always a continuous function. SDP is a

method designed to overcome these difficulties using two distinct ideas.

First, in the coordination problem, sensitivities are not used. Instead, coordination is performed using an approximate problem (for instance, a quadratic program) that uses all constraints and all variables in the original problem. This technique was originally used by Vanderplaats et al.[13] so that changes in constraint activity in the subproblems are accounted for and the feasible space of the model problem is more likely to contain points that will correctly identify the active constraint set.

Second, when near a solution, decomposition, i.e., solving subproblems separately, is not performed. A specific test for determining when $\boldsymbol{x}^k$ is near the solution is discussed subsequently. An SDP algorithm is any algorithm that fits the following form.

1) Use of decomposition to solve subproblems: if not near the solution, hold the linking variables constant at $\boldsymbol{x}_0 = \boldsymbol{x}_0^k$ and find a minimizer $\boldsymbol{x}_j^{\dagger k}$ for each subproblem $j = 1, \ldots, p$.

2) Coordination: using an approximate problem of the nonlinear program, which includes all of the constraints and all of the variables in the original problem, find a satisfactory next point.

3) Continue until some convergence criteria are met.

In particular, these ideas are now applied to the SQP algorithm of Wilson,[18] Han,[16] and Powell.[17]

To apply SDP to SQP, a few notes concerning notation, structure, and sparsity must be made. First, the Lagrangian [Eq. (14)] is expressed as a sum of terms [Eq. (15)]:

$$L(\boldsymbol{x}, \lambda, \mu) = f(\boldsymbol{x}) + \sum_i^{m_i} \lambda_i g_i(\boldsymbol{x}) + \sum_i^{m_e} \mu_i h_i(\boldsymbol{x}) = \sum_j^{p} L_j(\boldsymbol{x}_0, \boldsymbol{x}_j, \lambda_j, \mu_j) \tag{14}$$

$$L_j(\boldsymbol{x}_0, \boldsymbol{x}_j, \lambda_j, \mu_j) = f_j(\boldsymbol{x}_0, \boldsymbol{x}_j) + \sum_i^{m_{j,\text{ineq}}} \lambda_{ji} g_{ji}(\boldsymbol{x}) + \sum_i^{m_{j,\text{eq}}} \mu_{ji} h_{ji}(\boldsymbol{x}) \tag{15}$$

Each of the j terms represented in Eq. (15) corresponds to the jth subproblem of the form in Eq. (3). The Hessian of the Lagrangian $\boldsymbol{H}$ can also be represented as a sum of sparse Hessians $\boldsymbol{H}_j$ [as shown in Eq. (16)], each corresponding to a subproblem

$$\boldsymbol{H}_j = \nabla^2 L_j(\boldsymbol{x}_0, \boldsymbol{x}_j, \lambda_j, \mu_j) = \begin{bmatrix} \nabla_{\boldsymbol{x}_0 \boldsymbol{x}_0}{}^2 L_j & 0 & \nabla_{\boldsymbol{x}_0 \boldsymbol{x}_j}{}^2 L_j & 0 \\ 0 & 0 & 0 & 0 \\ \nabla_{\boldsymbol{x}_j \boldsymbol{x}_0}{}^2 L_j & 0 & \nabla_{\boldsymbol{x}_j \boldsymbol{x}_j}{}^2 L_j & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_{j00} & 0 & \boldsymbol{H}_{j0j} & 0 \\ 0 & 0 & 0 & 0 \\ \boldsymbol{H}_{jj0} & 0 & \boldsymbol{H}_{jjj} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{16}$$

The first subscripted index refers to the subproblem, and the second and third subscripts refer to derivatives with respect to the corresponding set of design problems. $\boldsymbol{B}_j^k$ is then the approximation of $\boldsymbol{H}_j$ during the kth iteration, $\boldsymbol{B}_{jjj}^k$ is the approximation of $\boldsymbol{H}_{jjj}$, and $\boldsymbol{B}^k = \sum \boldsymbol{B}_j^k$ is the approximation of $\boldsymbol{H}$. Step 1c is the Powell[17] safeguard for the BFGS update for the quadratic programming (QP) subproblems in SQP,[16–18] but it is applied to the approximate Hessian as a whole, so that elements corresponding to $\boldsymbol{x}_0$ are also approximated according to the quasi-Newton condition. SDP–SQP is defined as follows.

1) If during the past few outer iterations, the set of active constraints predicted by Eq. (23), i.e., the coordination calculation, has remained the same, then holding the linking variables constant, $\boldsymbol{x}_0 = \boldsymbol{x}_0^k$, find a minimizer $\boldsymbol{x}_j^{\dagger k}$ for each subproblem $j = 1, \ldots, p$ by repeating step 1a–1d. Start with $k_j = 1$, $\boldsymbol{x}_j^{1j} = \boldsymbol{x}_j^k$, and $\boldsymbol{B}_j^{1j} = B_j^k$.

a) At $\boldsymbol{x}_j^{kj}$ solve for a search direction $\boldsymbol{d}_j$ using the following quadratic program.

Minimize $\boldsymbol{d}_j \in \mathfrak{R}^{n_j}$:

$$\tfrac{1}{2} \boldsymbol{d}_j^T \boldsymbol{B}_{jjj}^{kj} \boldsymbol{d}_j + \nabla f_j(\boldsymbol{x}_0^k, \boldsymbol{x}_j^{kj}) \boldsymbol{d}_j \tag{17}$$

subject to

$$\nabla g_{ji}(\boldsymbol{x}_0^k, \boldsymbol{x}_j^{ki}) \boldsymbol{d}_j + g_{ji}(\boldsymbol{x}_0^k, \boldsymbol{x}_j^{kj}) \leq 0$$
$$\nabla h_{ji}(\boldsymbol{x}_0^k, \boldsymbol{x}_j^{kj}) \boldsymbol{d}_j + h_{ji}(\boldsymbol{x}_0^k, \boldsymbol{x}_j^{kj}) = 0$$

b) Set $\boldsymbol{x}_j^{kj+1} = \boldsymbol{x}_j^{kj} + \alpha \boldsymbol{d}_j$, where α is chosen to sufficiently decrease some appropriate merit function.

c) Update $\boldsymbol{B}_j^{kj+1}$ using

$$\delta = [\boldsymbol{x}_0^k \ \ldots \boldsymbol{x}_j^{kj+1} \ \ldots 0] - [\boldsymbol{x}_0^k \ \ldots \boldsymbol{x}_j^{kj} \ \ldots 0] \tag{18}$$

$$\boldsymbol{y}_j = \nabla L_j(\boldsymbol{x}_0^k, \boldsymbol{x}_j^{kj+1}, \lambda^k, \mu^k) - \nabla L_j(\boldsymbol{x}_0^k, \boldsymbol{x}_j^{kj}, \lambda^k, \mu^k) \tag{19}$$

$$\theta = \begin{cases} 1 & \text{if} \quad \delta^T \boldsymbol{y}_j \geq 0.2 \delta^T \boldsymbol{B}_j^{kj} \delta \\ \dfrac{0.8 \delta^T \boldsymbol{B}_j^{kj} \delta}{\delta^T \boldsymbol{B}_j^{kj} \delta - \delta^T \boldsymbol{y}_j} & \text{if} \quad \delta^T \boldsymbol{y}_j < 0.2 \delta^T \boldsymbol{B}_j^{kj} \delta \end{cases} \tag{20}$$

$$\boldsymbol{z}_j = \theta \boldsymbol{y}_j + (1 - \theta)(\boldsymbol{B}_j^{kj} \delta) \tag{21}$$

$$\boldsymbol{B}_j^{kj+1} = \boldsymbol{B}_j^{kj} + \frac{\boldsymbol{z}_j \boldsymbol{z}_j^T}{\boldsymbol{z}_j \delta} - \frac{(\boldsymbol{B}_j^{kj} \delta)(\boldsymbol{B}_j^{kj} \delta)^T}{\delta^T \boldsymbol{B}_j^{kj} \delta} \tag{22}$$

d) If convergence criteria are satisfied, set $\boldsymbol{x}_j^{\dagger k} = \boldsymbol{x}_j^{kj}$ and $\boldsymbol{B}_j^{\dagger k} = \boldsymbol{B}_j^{kj}$; otherwise repeat steps 1a–1c.

2) At $\boldsymbol{x}^{\dagger k} = [\boldsymbol{x}_0^k \ \boldsymbol{x}_1^{\dagger k} \ \ldots x_{n_p}^{\dagger k}]$ and $\boldsymbol{B}^{\dagger k} = \sum \boldsymbol{B}_j^{\dagger k}$, solve for a search direction $\boldsymbol{d}$ using the following quadratic program.

Minimize $\boldsymbol{d} \in \mathfrak{R}^n$:

$$\tfrac{1}{2} \boldsymbol{d}^T \boldsymbol{B}^{\dagger k} \boldsymbol{d} + \nabla f(\boldsymbol{x}^{\dagger k}) \boldsymbol{d} \tag{23}$$

subject to

$$\nabla g_i(\boldsymbol{x}^{\dagger k}) \boldsymbol{d} + g_i(\boldsymbol{x}^{\dagger k}) \leq 0$$
$$\nabla h_i(\boldsymbol{x}^{\dagger k}) \boldsymbol{d} + h_i(\boldsymbol{x}^{\dagger k}) = 0$$

3) Set $\boldsymbol{x}^{k+1} = \boldsymbol{x}^{\dagger k} + \alpha \boldsymbol{d}$, where α is chosen to sufficiently decrease some appropriate merit function.

4) For $j = 1, \ldots, p$, update $\boldsymbol{B}_j^{k+1}$ using Eqs. (18–22).

5) Repeat steps 1–4 until convergence.

SDP–SQP incorporates both elements required in the definition of an SDP algorithm. First, in step 2, coordination is performed using all variables and constraints. Second, in step 1, there is a test to ensure that the subproblems are not solved independently when $\boldsymbol{x}^k$ is near the solution. The test used in SDP–SQP requires the active constraint set to remain the same. Note that other possibilities exist (for example, that unit step lengths are accepted or that successive coordination steps are progressively shorter). It is expected that the test will be modified according to the application, as in the case of unconstrained minimization. The only requirement of the test is that as the algorithm converges, eventually the test should always be true.

Additionally, SDP–SQP assembles the coordination problem using estimates obtained during the solution process of the subproblems. This is accomplished by expressing the Lagrangian as a sum of terms [as in Eq. (14)] and updating a sparse approximation for each term. The idea stems from the work of Griewank and Toint,[19] who have shown that for the unconstrained case the update defined by step 1c retains the null space of $\boldsymbol{B}_j^k$. For the case at hand, retaining the null space retains the sparsity pattern shown in Eq. (16). To the authors' knowledge, this has not been applied to the constrained case, but it is hoped that the use of Eqs. (18–22) will provide for better Hessian estimates in fewer iterations.

IV. Proof of Global Convergence

Luenberger[20] has proven that SQP or modified quasi-Newton methods converge via a descent sequence to a point $\boldsymbol{x}*$ that satisfies the first-order Karush–Kuhn–Tucker (KKT) conditions. For simplicity, the proof assumes only inequality constraints, but the extension to equality constraints is straightforward. The proof consists of two theorems. The first theorem establishes the limit of any convergent subsequence of points generated by SQP is a solution, provided that a suitable merit function exists. The second theorem shows that the absolute value penalty function is a suitable merit function.

The new third theorem presented here establishes that points generated during the course of the subproblems also represent a descent sequence of the same merit function, so that $\{\boldsymbol{x}^1, \boldsymbol{x}^{\dagger 1}, \boldsymbol{x}^2, \boldsymbol{x}^{\dagger 2}, \boldsymbol{x}^3, \boldsymbol{x}^{\dagger 3}, \ldots\}$ is a descent sequence, thus proving that SDP–SQP converges to a point satisfying first-order KKT optimality conditions through a descent sequence.

Theorem 1: Global Convergence Theorem.[19] Let $A(\boldsymbol{x})$ be an algorithm on some space χ, and suppose that, given $\boldsymbol{x}^1$, the sequence $\{\boldsymbol{x}^k\}_{k=1}^{\infty}$ is generated satisfying $\boldsymbol{x}^{k+1} \in A(\boldsymbol{x}^k)$.

Let a solution set Γ contained in the space χ be given, and suppose 1) all points $\boldsymbol{x}^k$ are contained in a compact set S, which is also contained in χ; 2) there is a continuous function P on χ such that if $\boldsymbol{x} \notin \Gamma$ then $P(\boldsymbol{y}) < P(\boldsymbol{x})$ for all $\boldsymbol{y} \in A(\boldsymbol{x})$, and if $\boldsymbol{x} \in \Gamma$ then $P(\boldsymbol{y}) \leq P(\boldsymbol{x})$ for all $\boldsymbol{y} \in A(\boldsymbol{x})$; and 3) the mapping A is closed at points outside Γ. Then the limit of any convergent subsequence of $\{\boldsymbol{x}^k\}$ is a solution.

Theorem 2: Use of the Absolute Value Penalty Function.[19] Let $\boldsymbol{d}(\boldsymbol{d} \neq 0)$ be a solution of the quadratic program [Eq. (17)] and λ be the vector of multipliers associated with the constraints. Then if $c > \max\{\mu_j\}$ the vector $\boldsymbol{d}$ is a descent direction for the penalty function,

$$P(\boldsymbol{x}) = f(\boldsymbol{x}) + c \sum_{i}^{m_{\text{ineq}}} g_i^+(\boldsymbol{x}) \tag{24}$$

where $g^+ \equiv \max\{0, g\}$.

Theorem 3: Solutions to Subproblems Are Also Descent Directions. Let $\boldsymbol{d}_j$, $(\boldsymbol{d}_j \neq 0)$ be a solution of the quadratic program used in the solution process of the subproblems [Eq. (23)] and λ_j be the vector of multipliers associated with the constraints. As a convention, let $\boldsymbol{d} = [0 \ldots \boldsymbol{d}_j \ldots 0]$. Then if $c > \max\{\lambda_{ji}\}$, the vector $\boldsymbol{d}$ is a descent direction for the same absolute value penalty function stated in Eq. (24).

Proof of Theorem 3. Let $I_{j,\text{ineq}}(\boldsymbol{x})$ denote the indices for the set of inequality constraints for the jth subproblem that are violated at the point $\boldsymbol{x}$. The penalty function P is written as a function of the scalar α,

$$P(\boldsymbol{x} + \alpha\boldsymbol{d}) \equiv f(\boldsymbol{x} + \alpha\boldsymbol{d}) + c \sum_{i}^{m_{\text{ineq}}} g_i^+(\boldsymbol{x} + \alpha\boldsymbol{d}) \tag{25}$$

Note that all of the constraints are used in Eq. (25) and that $f = \sum f_j$. Note also that all functions (objective and constraint) that are not in the subproblem will not change in value because they do not depend on $\boldsymbol{x}_j$:

$$f(\boldsymbol{x} + \alpha\boldsymbol{d}) = f(\boldsymbol{x}) - f_j(\boldsymbol{x}) + f_j(\boldsymbol{x} + \alpha\boldsymbol{d}) \tag{26}$$

$$\text{if} \quad J \neq j \quad \text{then} \quad g_{Ji}(\boldsymbol{x} + \alpha\boldsymbol{d}) = g_{Ji}(\boldsymbol{x}) \tag{27}$$

$$P(\boldsymbol{x} + \alpha\boldsymbol{d}) = f(\boldsymbol{x}) - \alpha\nabla f_j(\boldsymbol{x})\boldsymbol{d} + c\left[\sum_{i}^{m_{j,\text{ineq}}} (g_{ji}(\boldsymbol{x}) + \alpha\nabla g_{ji}(\boldsymbol{x})\boldsymbol{d})^+ + \sum_{J\neq j}^{p} \sum_{i}^{m_{J,\text{ineq}}} g_{Ji}^+(\boldsymbol{x})\right] + \mathcal{O}(\alpha) \tag{28}$$

In Eq. (28), the notation $\mathcal{O}(\alpha)$ means that the remaining terms are no larger than a linear function of α. Two of the first-order KKT conditions for the solution of the quadratic approximation for the subproblem [Eq. (17)] are

$$\nabla f_j(\boldsymbol{x}) = -\boldsymbol{B}_{jjj}\boldsymbol{d}_j + \sum_{i}^{m_{j,\text{ineq}}} \lambda_{ji}\nabla g_{ji}(\boldsymbol{x}) \tag{29}$$

$$\nabla g_{ji}(\boldsymbol{x})\boldsymbol{d}_j \leq -g_{ji}(\boldsymbol{x}) \tag{30}$$

allowing the relations

$$\sum_{i\in I_{j,\text{ineq}}} \nabla g_{ji}(\boldsymbol{x})\boldsymbol{d} \leq \sum_{i}^{m_{j,\text{ineq}}} -g_{ji}^+(\boldsymbol{x}) \tag{31}$$

$$\nabla f_j(\boldsymbol{x})\boldsymbol{d} \leq -\boldsymbol{d}_j^T\boldsymbol{B}_{jjj}\boldsymbol{d}_j + \max\{\lambda_{ji}\}\sum_{i}^{m_{j,\text{ineq}}} g_{ji}^+(\boldsymbol{x}) \tag{32}$$

Substituting Eqs. (31) and (32) into Eq. (28) gives

$$P(\boldsymbol{x} + \alpha\boldsymbol{d}) \leq P(\boldsymbol{x}) + \alpha\left[-\boldsymbol{d}_j^T\boldsymbol{B}_{jjj}\boldsymbol{d}_j + (c - \max\{\lambda_{ji}\})\sum_{i}^{m_{j,\text{ineq}}} g_{ji}^+(\boldsymbol{x})\right] + \mathcal{O}(\alpha) \tag{33}$$

Because $\boldsymbol{B}_{jjj}$ is constructed to be positive definite and $c > \max\{\lambda_j\}$, it follows that for a sufficiently small α,

$$P(\boldsymbol{x} + \alpha\boldsymbol{d}) \leq P(\boldsymbol{x}) \tag{34}$$

Thus, each subproblem reduces the penalty function.

V. Comparison by Example

As a small example, both SQP and SDP–SQP are applied to a scaled version of the NLP in Eq. (4) The variable transformations are

$$\begin{aligned}[x_{01}, x_{02}] &= [R, 0.1L]\\ [x_{11}] &= [10t_s]\\ [x_{21}] &= [t_h]\end{aligned} \tag{35}$$

The SQP algorithm used is the routine constr in the MATLAB® optimization toolbox,[21] and the SDP–SQP routine was also written in MATLAB. The collection of objective functions and constraints for $j = 0$ is

$$f_0 = 0$$

$$g_{00} = 1 - \frac{x_{01}^2 x_{02}}{413{,}000} \leq 0 \tag{36}$$

$$g_{01} = x_{02} - \frac{1.0}{0.0417} \leq 0$$

The two subproblems appear as Eq. (37) (subproblem 1) and Eq. (38) (subproblem 2).

Minimize $x_{11} \geq 1$:

$$f_1 = 0.662x_{01}x_{02}x_{11} + 0.0158x_{02}x_{11}^2 + 0.1984x_{01}x_{11}^2 \tag{37}$$

subject to

$$g_{11} = 0.0193x_{01} - t_s \leq 0$$

Minimize $x_{21} \geq 0.1$:

$$f_2 = 1.777x_{01}^2x_{21} \times 10^{-4} \tag{38}$$

subject to

$$g_{21} = 0.131x_{01} - x_{21} \leq 0$$

Both algorithms use the starting point $\boldsymbol{x} = [60, 20, 40, 15]$. The iteration history is shown as a function of the outer iteration in Fig. 5.

SDP–SQP used decomposition for the first two iterations and converged upon the solution in a total of 8 iterations, compared to SQP, which performed 11 iterations. Additionally, SDP–SQP evaluated subproblem 0 8 times, subproblem 1 10 times and subproblem 2 12 times, whereas SQP made 11 evaluations of all three subproblems. CPU time was not measured for this example, as no increase in speed is expected. In fact, benefits in speed are not expected unless the time for a function call exceeds the time for solving a quadratic program and the time for solving a large quadratic program (the coordination problem) greatly exceeds the time for solving a small quadratic program (the subproblems).

The example problem is not of the scale intended for SDP–SQP. However, the results shown here are promising, and larger applications may show significant benefit.

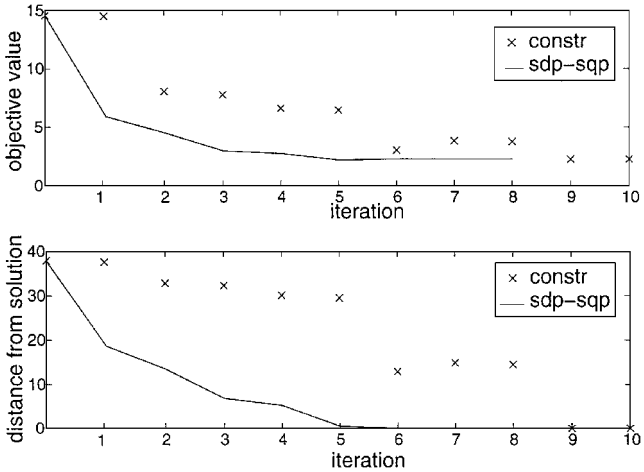

Fig. 5 Iteration history of the SDP–SQP and plain SQP algorithm. Distance from the solution is measured by the Euclidean norm in the *x* space with the norm measured just prior to the execution of the line search so that the improvement in the SDP–SQP norm reflects the use of the subproblems.

VI. Open Issues

This presentation examined the general definition of SDP and its application to SQP. As this investigation is ongoing, the open issues concerning both SDP and SDP–SQP are addressed separately.

A. SDP

On reviewing Sec. II, it is pertinent to wonder whether decomposition is computationally useful. To this end, a few comments should be made.

First, SDP is defined as a two-phase process, and the number of iterations in the initial phase (which uses decomposition to independently solve the subproblems) is not known a priori, so further computational experience with SDP is necessary. If measured in computation time, the length of time spent in the first phase determines the benefit of SDP. However, the transition to the second phase (where decomposition is not directly used) can be based on any test that will eventually become true during the course of the algorithm.

Second, even if the first phase does not last long enough to warrant the effort of implementing SDP when measured in terms of function evaluation and computational time, it is possible to tailor the algorithm such that the coordination problem uses approximations compiled during the solution process of the subproblems. For example, the coordination problem in SDP–SQP uses Hessian estimates that were acquired by solving the subproblems using an SQP algorithm. In fact, using such approximations may actually increase the robustness because the approximations used during coordination would be more accurate.

Third, optimization at the component level is already widely used in industry. Through years of experience, these optimization routines have been finely tuned and their performance is well understood. A hierarchically decomposed method such as SDP can integrate already used design optimization tools without significantly reworking and recoding the system model or the optimization algorithm.

Fourth, the human aspect of engineering requires understanding the solution of any engineering design problem beyond the mathematics and computations used. If the problem has been decomposed, the interaction between subproblems may be understood better by examining the behavior of the individual subproblems. Indeed, this may be the only way for an individual to understand the behavior of truly complex engineering design problems.

B. SDP–SQP

Defining the approximation of the Hessian as a sum of sparse matrices is not new. The original definition used in Eqs. (18–22) stems from a combination of work by Broyden,[22] Fletcher,[23] Goldfarb,[24] Shanno,[25] Powell,[17] and Greiwank and Toint.[19] However, theoretical works by Greiwank and Toint establishing local convergence rate for unconstrained optimization problems assume that the underlying functions are convex on $\Re^n$. Hopefully, this can be relaxed so the point where L_j need only be convex on $\Re^{n_j}$ and $L = \sum L_j$ is strictly locally convex near the solution on $\Re^n$. Understanding the convexity requirements of L_j will lead to a local convergence analysis theorem.

Furthermore, there is very little experience with applying SDP to practical engineering problems where issues such as parallelism, computation time, and the ratio of function call time to QP solution time and problem size can be studied more thoroughly.

VII. Conclusion

The graph partitioning methods developed for model-based decomposition of optimal designs have removed much of the ad hoc process that characterized earlier decomposition approaches. Many different decomposition forms can be discovered formally based on rigorous methodologies. The actual solution of decomposed NLP problems has been shown to present important convergence difficulties. This article attempted to place these difficulties in a formal context and to define algorithms for hierarchical NLPs that possess desirable convergence properties while remaining simple extensions of existing algorithms for general NLPs.

Global convergence of an SDP–SQP algorithm is straightforward, and the local convergence properties are currently under further investigation.

Acknowledgments

This research has been partially supported by the Automotive Research Center at the University of Michigan, a U.S. Army Center of Excellence in Modeling and Simulation of Ground Vehicles, under Contract DAAE07-94-C-R094. This support is gratefully acknowledged. Additionally, the authors are grateful to Spillios Fassois, Antony Kontos, and the University of Patras for providing an excellent research atmosphere for the first author and supporting the sabbatical of the second author.

References

[1]Conn, A. R., Gould, N. I. M., and Toint, P. L., *LANCELOT, A Fortran Package for Large-Scale Nonlinear Optimization*, Springer–Verlag, Berlin, 1992, pp. 1–13.

[2]Papalambros, P. Y., "Optimal Design of Mechanical Engineering Systems," *Journal of Mechanical Design*, Vol. 117, June 1995, pp. 55–62.

[3]Lootsma, F. A., "Exploitation of Structure in Non-Linear Optimization," *Parallel Computing 89*, edited by D. J. Evans, G. R. Joubert, and F. J. Peters, North-Holland, New York, 1990, pp. 31–45.

[4]Dantzig, G. B., and Wolfe, P., "Decomposition Principle for Linear Programs," *Operations Research*, Vol. 8, Jan.–Feb. 1960, pp. 101–111.

[5]Azarm, S., and Li, W., "A Two Level Decomposition Method for Design Optimization," *Engineering Optimization*, Vol. 13, No. 3, 1988, pp. 211–224.

[6]Azarm, W., and Li, W., "Optimality and Constrained Derivatives in Two-Level Design Optimization," *Journal of Mechanical Design*, Vol. 112, Dec. 1990, pp. 563–568.

[7]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.

[8]Wagner, T. C., and Papalambros, P. Y., "A General Framework for Decomposition Analysis in Optimal Design," *Advances in Design Automation*, Vol. 2, 1993, pp. 315–325.

[9]Michelena, N., and Papalambros, P. Y., "Optimal Model-Based Decomposition of Powertrain System Design," *Journal of Mechanical Design*, Vol. 117, No. 4, 1995, pp. 499–505.

[10]Beightler, C., and Phillips, D. T., *Applied Geometric Programming*, Wiley, New York, 1976, pp. 500–507.

[11]Wagner, T. C., "A General Decomposition Methodology for Optimal System Design," Ph.D. Dissertation, Dept. of Mechanical Engineering and Applied Mechanics, Univ. of Michigan, Ann Arbor, MI, May 1993.

[12]Thareja, R. R., and Haftka, R. T., "Efficient Single-Level Solution of Hierarchical Problems in Structural Optimization," *AIAA Journal*, Vol. 28, No. 3, 1990, pp. 506–514.

[13]Vanderplaats, G. N., Yang, Y. J., and Kim, D. S., "Sequential Linearization Method for Multilevel Optimization," *AIAA Journal*, Vol. 28, No. 2, 1990, pp. 290–295.

[14]Vanderplaats, G. N., and Yoshida, N., "Efficient Calculation of Optimum Design Sensitivity," *AIAA Journal*, Vol. 23, No. 11, 1985, pp. 1798–1803.

[15]Schittkowski, K., "The Nonlinear Programming Method of Wilson, Han, and Powell with an Augmented Lagrangian Type Line Search Function," *Numerische Mathematik*, Vol. 38, Oct. 1981, pp. 83–114.

[16]Han, S. P., "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," *Mathematical Programming*, Vol. 11, Dec. 1976, pp. 263–282.

[17]Powell, M. J. D., "The Convergence of Variable Metric Methods for Nonlinear Constrained Optimization Calculations," *Nonlinear Programming 3*, edited by O. L. Mangasarian, Academic, New York, 1978, pp. 27–64.

[18]Wilson, R. B., "A Simplicial Algorithm for Concave Programming," Ph.D. Dissertation, Graduate School of Business Administration, Harvard Univ., Cambridge, MA, 1963.

[19]Griewank, A., and Toint, P. L., "Partitioned Variable Metric Updates for Large Structured Optimization Problems," *Numerische Mathematik*, Vol. 39, June 1982, pp. 119–137.

[20]Luenberger, D. G., *Linear and Nonlinear Programming*, Addison–Wesley, Reading, MA, 1984.

[21]Grace, A., *MATLAB Optimization Toolbox Manual*, Math Works Inc., Natick, MA, 1994.

[22]Broyden, C. G., "The Convergence of a Class of Double-Rank Minimization Algorithms," *Journal for the Institute of Mathematics and Its Applications*, Vol. 6, No. 1, 1970, pp. 76–90.

[23]Fletcher, R., "A New Approach to Variable Metric Algorithms," *Computer Journal*, Vol. 13, No. 3, 1970, pp. 317–322.

[24]Goldfarb, D., "A Family of Variable Metric Methods Derived by Variational Means," *Mathematics of Computation*, Vol. 24, No. 109, 1970, pp. 23–26.

[25]Shanno, D. F., "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computation*, Vol. 24, No. 111, 1970, pp. 647–657.

A. D. Belegundu
Associate Editor