

## TRENDS AND CHALLENGES IN SYSTEM DESIGN OPTIMIZATION

**Panos Y. Papalambros**  
**Nestor F. Michelena**  
Optimal Design Laboratory  
Department of Mechanical Engineering  
University of Michigan  
Ann Arbor, Michigan, USA

### ABSTRACT

Design optimization is now a mainstream discipline in high technology product development and a natural extension of the ever-increasing analytical abilities of computer-aided engineering. Business factors, such as globalization, outsourcing, supply-chain management, and rapid new product deployment, are placing increased emphasis on a “systems” approach to product design. Technological factors, such as the emergence of new technologies at the intersection of traditional ones (e.g., MEMS, nanotechnologies, and biotechnologies) and the widespread use of the internet, are also forcing increased emphasis on the same “systems” approach. As a result, there has been an increased need to study design optimization methods that can address effectively the “system” problem. In this article we view a system as a collection of entities that might be properly structured so that some form of decomposition is possible. The problem has two parts: finding a proper system partition that captures an appropriate structure, and solving the partitioned problem with a coordination method that guarantees some form of convergence that is meaningful for the original undecomposed problem. We review some key ideas in these two issues and we show how they can be used effectively in product development processes, such as target cascading, product family design, and combined design and control of “smart” artifacts.

### 1 INTRODUCTION

Perhaps one of the few agreed-upon characterizations of the world today is its “complexity.” This complexity is nothing new, but the present emphasis stems from an often undeclared desire to deal with it directly. This desire in turn stems from our increased ability to deal with the complexity of the physical world, to a large degree due to the rapid growth of computing and information technology. The same trend is seen in the design of engineered artifacts. Here complexity is manifested by a gradual migration of efforts towards a “system” design. Occasionally the term “system” can be rigorously defined within a specific discipline, for example, an “input-output” system where the system is a function mapping a set of input quantities to a set of output quantities. The term is also used more casually in a variety of contexts.

Often the term “system” is used in contrast to the term “component.” The context then is to address complexity derived from studying a collection of components that function jointly thus comprising a system. A system can be composed of other systems that are often referred as “subsystems,” if one wants to emphasize the indivisible nature of a component. For example, a system can be an automobile—a collection of a great number of subsystems and components. A collection of variants of automobiles can be another definition of a system. This is one of the contexts that the term “system” is used in the present article.

Another frequent use of the term “system” is to suggest a more encompassing viewpoint—the “system view.” Here the context is that an engineered artifact must be studied from various perspectives *simultaneously*, the assumption being that the artifact’s functions may seem different from each perspective but their interactions are critical to the overall function of the artifact. Complexity is induced from the need to account for all these perspectives together. For example, the design of controllable artifacts requires that the embodiment design and controller design be studied simultaneously in order to produce efficient smart products. This is another context that “system” is used in the present article.

Design optimization assumes a decision-making paradigm for the design process. The formal *mathematical model* of the optimization problem is a statement of the form

$$\begin{aligned}
& \text{minimize } f(\mathbf{x}) \\
& \text{subject to } \mathbf{h}(\mathbf{x}) = \mathbf{0} \\
& \quad \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
& \quad \mathbf{x} \in \mathcal{X} \subseteq \mathfrak{R}^n
\end{aligned} \tag{1}$$

where the scalar *objective function*  $f(\mathbf{x})$  provides the comparison criterion among different alternatives, the vector-valued functions  $\mathbf{h} = (h_1, h_2, \dots, h_m)^T$  and  $\mathbf{g} = (g_1, g_2, \dots, g_{m_2})^T$  are the *constraint functions* that determine whether a design is feasible, and  $\mathbf{x}$  is the  $n$ -dimensional vector of the *design variables*, where  $n$  is finite. In many embodiment design problems the design variables take continuous real values in the  $n$ -dimensional real space  $\mathfrak{R}^n$ . However, in many problems design variables may take only discrete values, such as standard sizes of cross-sections or configuration design problems. In model (1) the type of values used is described by the set  $\mathcal{X}$ , the *set constraint*. Models with continuous variables are generally easier to solve with techniques based on differential calculus.

Design problems often include two or more competing objectives, leading to the *multiobjective* or *multicriteria* problem

$$\begin{aligned}
& \text{minimize } \mathbf{c}(\mathbf{x}) \\
& \text{subject to } \mathbf{h}(\mathbf{x}) = \mathbf{0} \\
& \quad \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
& \quad \mathbf{x} \in \mathcal{X} \subseteq \mathfrak{R}^n
\end{aligned} \tag{2}$$

where  $\mathbf{c}$  is the vector of  $I$  real-valued criteria  $c_i$ . The feasible values for  $\mathbf{c}(\mathbf{x})$  constitute the *attainable set*  $A$ . The multicriteria formulation is converted into a *scalar substitute problem* of the general form

$$\begin{aligned}
& \text{minimize } f = \sum_i w_i f_i(w_i) f_2(c_i, \mathbf{m}_i) \\
& \text{subject to } \mathbf{h}(\mathbf{x}) = \mathbf{0} \\
& \quad \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
& \quad \mathbf{x} \in \mathcal{X} \subseteq \mathfrak{R}^n
\end{aligned} \tag{3}$$

where the scalars  $w_i$  and vectors  $\mathbf{m}_i$  are preference parameters. Pareto optimality is a common preference structure. A point in the design space is a Pareto (optimal) point if there exist no feasible point that would reduce one criterion without increasing the value of one or more of the other criteria. The functions  $f$ ,  $\mathbf{h}$ , and  $\mathbf{g}$  can be explicit algebraic expressions but may also be the formal statement of a complex procedure involving internal calculations and realized only as a computer program—what is often called a *simulation model*, such as numerical solutions of coupled differential equations. An *analysis model* refers to computing the functions  $f$ ,  $\mathbf{h}$ , and  $\mathbf{g}$ . An (*optimal*) *design model* refers to Eq. (1). If the functions represent algebraic or equivalent relations of a finite design vector  $\mathbf{x}$ , then model (1) represents a *mathematical programming problem*. If differential or integral operators are involved and the variables  $x_i = x_i(t)$ ,  $t \in \mathfrak{R}$ , are defined in an *infinite dimensional* space, then we have a *variational problem*.

Designing complex engineered artifacts (and collections of them) most likely requires use of simulations coupled with optimization techniques. As complexity increases, our ability to employ intuition (even for understanding the computed trade-offs) declines rapidly. Furthermore, our ability to actually solve these system optimization problems becomes suspect as dimensionality increases. An approach towards solving these challenging problems is obvious: try to break down the problem to smaller ones that can be solved more easily and then compose the overall system solution from the solution of its parts. This decomposition approach can be effective but its practical and rigorous implementation is much more difficult than one might expect.

In this article we will first explore the issues associated with decomposition strategies for optimal system design. Then we will look at some specific implementations representative of current research efforts in this area. More specifically, we will briefly examine work on product development processes, such as platform design and target cascading, as well as

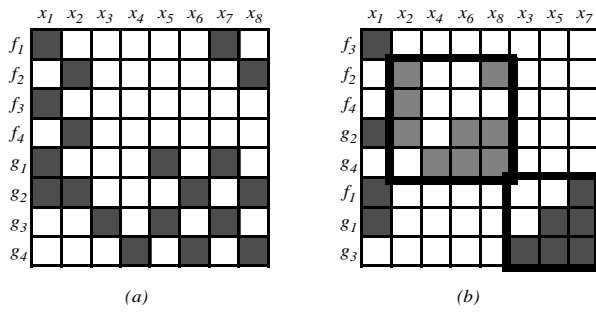


Figure 1. (a) Original form of an FDT and (b) derived form after reordering rows and columns to identify sub-problems

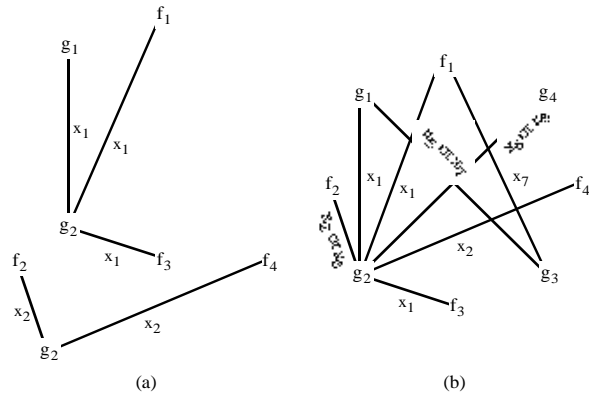


Figure 2. (a) Tree representation of function dependency on variables  $x_1$  and  $x_2$ , and (b) undirected graph representation of problem of Figure 1

on simultaneous design and control strategies. Our aim here is to scope out some key ideas rather than present a comprehensive review. Thus references are relatively sparse and biased towards the work experiences of the authors.

A decomposition strategy is a process with two steps: partitioning and coordination. Partitioning the problem into smaller subproblems was originally done in an ad hoc manner. Newer developments allow us to do this step in a rigorous and flexible manner. Coordination of the solutions of the individual subproblems so that the solution of the overall problem is achieved is the core activity in a decomposition strategy. Coordination is difficult to perform rigorously and efficiently, and eventual success depends on the form of the problem partitioning. We will explore these issues in the next two sections.

## 2 PARTITIONING: BREAKING A PROBLEM DOWN TO PIECES

Partitioning methods are commonly classified as *object decomposition* (by physical components), *aspect decomposition* (by knowledge domains—the original motivation for multidisciplinary optimization or MDO), *sequential decomposition* (by directed flow of elements or information), and *model-based*. Object and aspect decomposition assume a “natural” decomposition of the problem. However, drawing “boundaries” around physical components and subassemblies is subjective, while division by specialties (knowledge domains) may be dictated by management considerations that fail to account for disciplinary coupling. Sequential decomposition presumes unidirectionality of design information flow that may contradict the cooperative behavior desirable in concurrent engineering. Model-based partitioning uses the mathematical functional representation of design objectives and constraints in the model(s) to identify unconnected or weakly-connected structures implicit in the mathematical design model, and is limited to the design decisions included in the model. However, the partitioning process can be automated and various ways to break down the problem can be quickly generated and evaluated.

The process starts with establishing the problem’s design function dependency on design variables, represented by a Boolean matrix termed the *functional dependence table* (FDT). Rows are labeled with relation/function names and columns are labeled with variable names. The entry in the  $i$ th row and  $j$ th column is “true” if the  $i$ th function depends on the  $j$ th variable; otherwise, it is “false.” A typical FDT is shown in Figure 1(a), the shaded boxes indicating a “true” Boolean value. Figure 1(b) shows the FDT for the same problem after  $x_1$  has been selected as the linking variable, and rows and columns have been reordered to reveal two partitions of the problem: subproblem 1 with functions  $\{f_2, f_4, g_2, g_4\}$  and local variables  $\{x_2, x_4, x_6, x_8\}$ , and subproblem 2 with functions  $\{f_1, g_1, g_3\}$  and local variables  $\{x_3, x_5, x_7\}$ . (A hierarchical coordination strategy would consider a master problem on the linking variable  $x_1$  and function  $f_3$ .)

The partitioning above can be made rigorous by modeling the decomposition problem as a network optimization problem (Michelena and Papalambros 1995). Mathematical relations are modeled as processing units of a communication network and design and state variables are communication links between these units. The optimal decomposition problem is then formulated as one of finding the communication links whose failure would reduce network reliability the most.

Extending this partitioning model to include other graph metrics, such as keeping subsystem sizes balanced and number

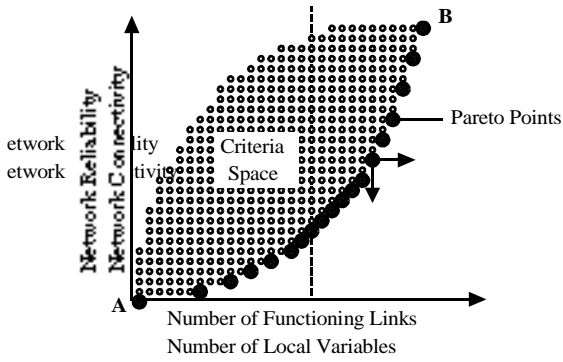


Figure 2. Pareto solution of partitioning problem

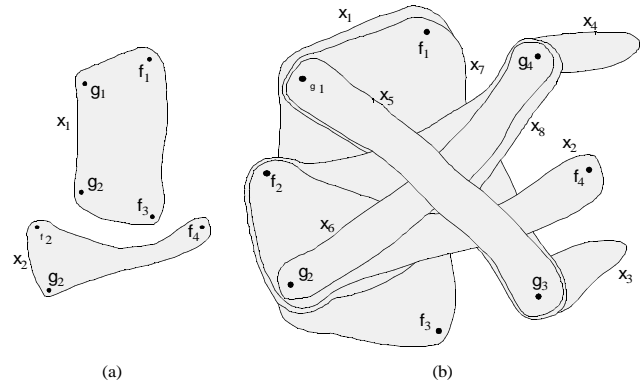


Figure 4. (a) Hyperedge representation of function dependency on variables  $x_1$  and  $x_2$ , and (b) hypergraph representation of problem of Figure 1

of linking variables small, has led to partitioning methods that use hypergraph or integer programming formulations (Michelena and Papalambros 1997; Krishnamachari and Papalambros 1997).

Details of the network reliability, hypergraph partitioning, and integer programming formulations of problem partitioning follow.

### Network reliability approach to problem partitioning

The partitioning problem is formulated as a multiobjective optimization problem with two conflicting objectives, namely, minimize the number of linking variables and minimize the size of the subproblems by maximizing the number of partitions. A design problem (and therefore its FDT) can be represented by an undirected, linear graph. This representation does not require a priori knowledge of input-output relations or causality between variables. A simplified version of the graph representation employed by Wagner and Papalambros (1993) is used in the network reliability formulation of problem partitioning. Wagner and Papalambros assigned a clique to each variable, and proved the equivalence of disjoint partitions in the FDT and connected components in its graph representation. In a clique-based representation, a clique connects vertices representing functions that depend on the same variable. Thus, if  $k$  functions depended on a variable, a  $k$ -clique was constructed for such a variable. In the network formulation a tree is used to connect functions that depend on the same variable. This tree-based representation is sufficient for the purpose of identifying connected components. An edge is labeled with the variable name(s) the functions associated with the edge's incident vertices depend on. Figure 2(a) shows trees associated with variables  $x_1$  and  $x_2$  in the FDT of Figure 1. Figure 2(b) shows the graph representation for the entire problem of Figure 1.

The undirected graph representation motivates formulating partitioning as network reliability optimization with two conflicting objectives: maximize the number of functioning links and minimize a measure of overall network reliability. Functioning links are identified with local variables, while failed links are identified with linking variables. Common sense indicates that the more connected a network is, the more reliable it is. This network optimization may be also viewed as selecting critical communication links, the linking variables, and assigning their control to a top decision-maker, the *master problem*. The control of the other links, the local variables, is left to low-level decision-makers, the *subproblems*. Critical links are those whose failure lessens the most the overall reliability of the network. Critical links are identified by finding the Pareto points shown in Figure 3. Solution point "A" corresponds to the case where every variable is considered a linking variable, so the problem is entirely disconnected. Solution point "B" corresponds to the case where every variable is considered a local variable, so the problem is maximally connected.

Two measures, all-terminal network state and network resilience, can be used as measures of network—and therefore design problem—connectivity. The all-terminal state  $\Phi_A$  of a network is one for totally connected graphs and zero for disconnected graphs, independent of the relative size of the parts. Network resilience  $\Phi_{PC}$  denotes the number of pairs of vertices in a network that are connected. Network resilience allows identifying partitions of similar size since a low value indicates that the network is divided in many pieces of similar size. Efficient algorithms to compute these two measures of network connectivity were presented in Michelena and Papalambros (1995). A multiobjective problem with two conflicting objectives is thus formulated: (i) minimize the number of linking variables, i.e., failed links, by maximizing

the sum of the indicator variables  $e_i$  ( $e_i$  is one if edge  $i$  is functioning and zero otherwise), and (ii) maximize the number of partitions by minimizing either the all-terminal network state or the network resilience.

$$\text{minimize } \left\{ - \sum_{i=1}^m e_i, \phi_A(e) \text{ or } \phi_{PC}(e) \right\} \quad (4)$$

$$e \in \{0, 1\}^m$$

Since  $\phi_A$  is zero or one, Pareto points  $e^*$  are generated by factoring of  $\phi_A$  and identifying factors that take a zero value with the fewest number of indicators  $e_i$  equal to zero. For  $\phi_{PC}$ ,  $m$  Pareto points  $e^*$  can be found by minimizing  $\phi_{PC}$  subject to the successive constraints  $\sum_{i=1}^m e_i \leq m, m-1, \dots, 2, \text{ or } 1$ , over the space  $\{0,1\}^m$ . Alternatively, as shown in Michelen and Papalambros (1995), a method of objective weighting or a greedy algorithm can be used to generate the Pareto solutions.

### Hypergraph partitioning approach to problem partitioning

The partitioning problem can be also formulated as a hypergraph partitioning problem. The resulting representation is robust enough to account for computational demands of the modules (functions) in the model and of the strength of their interdependencies. This approach makes use of recent advances common to such diverse areas as graph theory, VLSI design, computational mechanics, and parallel computing.

A design problem can be represented by a hypergraph  $H = (V, E_H)$  in which hyperedges in  $E_H$  are subsets of  $V$ . Vertices in  $V$  represent design relations (i.e., objective and constraints), and hyperedges represent design and intermediate variables. A hyperedge  $e_i \in E_H$  represents a variable  $x_i$  if and only if for every vertex  $v_i \in e_i$ , the function associated with  $v_i$  depends on  $x_i$ . Figure 4(a) shows hyperedges associated with variables  $x_1$  and  $x_2$  in the FDT of Figure 1(a), respectively. Figure 4(b) shows the hypergraph representation for the entire problem of Figure 1. Optimal partitioning calls for (i) minimizing the interconnection between subproblems and (ii) balancing the size of the subproblems. The former is aimed at reducing the effort to coordinate individual subproblems, and the latter is aimed at matching available computational resources. Hence, the following hypergraph partitioning problem can be formulated (Michelen and Papalambros 1997):

*Hypergraph K-Partitioning Problem.* Given a hypergraph  $H = (V, E_H)$  containing  $N$  vertices  $V = \{v_1, v_2, \dots, v_N\}$  with positive weights  $w_v(v_i)$ , and  $M$  hyperedges  $E_H = \{e_1, e_2, \dots, e_M\}$  with positive weights  $w_e(e_j)$ , a constant  $2 \leq K \leq N$ , and a partition load (or size) vector  $\mathbf{m} = (m_1, \dots, m_K)$  such that  $m_k \leq m_{k+1}$  and  $\sum_{k=1}^K m_k = \sum_{i=1}^N w_v(v_i)$ , find a partition of  $V$  into  $K$

disjoint subsets  $P^K = \{V_1, V_2, \dots, V_K\}$  that minimizes (i) the total weight of the hyperedges cut by  $P^K$ ,  $C(P^K)$ , and (ii)

$$\left| \sum_{v \in V} w_v(v_i) - m_k \right| \text{ for every } k \text{ in } \{1, 2, \dots, K\}. \text{ The hyperedges cut by } P^K \text{ are } E_H^C(P^K) = \{e_j \in E_H \text{ such that there exist } v_{i1}, v_{i2} \text{ in } e_j, v_{i1} \in V_{j1} \in P^K, v_{i2} \in V_{j2} \in P^K, \text{ and } j_1 \neq j_2\}. \text{ Thus, the total weight of the hyperedges cut by } P^K \text{ is}$$

$$C(P^K) = \sum_{e \in E^C(P^K)} w_e(e_j).$$

When this formulation is applied to design problem partitioning, vertex weights represent computational costs (e.g., CPU time or memory) for the design relations, edge weights depict coupling strength or amount of transferred data between computational (e.g., simulation) modules, and partition loads represent processing capabilities in a distributed computational environment. Solution methods for the hypergraph  $K$ -partitioning problem include iterative improvement partitioning methods, such as the Kernighan-Lin algorithm, and global partitioning methods, such as spectral algorithms.

### Integer programming approach to problem partitioning

Hierarchical partitioning of a design problem has also been formulated as an integer program (Krishnamachari and

Papalambros 1997). In the proposed “hierarchical decomposition synthesis” methodology, a hierarchically decomposed optimal design problem is obtained from studying first a general design problem (GDP) that contains only design relations but has no objective(s) defined. As in the hypergraph partitioning formulation, the integer linear programming (ILP) formulation assumes that there are two desirable characteristics of the decomposed design problem. The subproblems should be relatively small in size to facilitate comprehension and computation, and of approximately the same size to facilitate validation, parametric studies, and load balancing in case of parallel solution. The first characteristic leads to an objective function that attempts to minimize both the size of the master problem and the average size of the subproblems: minimize  $w_m$  (size of the master problem) +  $w_s$  (average size of the subproblems), where  $w_m, w_s$  are weights. The second characteristic imposes the constraint:  $K_s \times$  (size of the smallest subproblem)  $\leq$  (size of the largest subproblem), where  $K_s \geq 1$  is a size factor. The size of a problem is defined as equal to the sum of the number of variables and the number of design functions that it contains.

In the formulation, each problem is designated by its variables and functions. All variables and functions in the problem must be assigned to the master problem or some subproblem. The master problem contains design relations that are functions exclusively of the linking variables. A local variable belongs to a subproblem if the function that depends on that variable is in the subproblem. Each function can belong to only one subproblem. Two different functions belonging to two different subproblems cannot have any common variables other than the linking ones. An integer linear programming (ILP) model is created whose zero-one variables indicate what problem the design variables and functions are assigned to. The model is kept linear by assuming that the number of subproblems  $K$  is fixed during optimization. The ILP model is advantageous since it represents a difficult but well-studied optimization problem. The global optimum may be found using branch and bound or cutting plane methods, and a lower bound on the ILP solution can be always obtained by solving the relaxed continuous LP (see, e.g., Papadimitriou and Steiglitz 1982). In the demonstration examples in Krishnamachari and Papalambros (1997) the models are first represented using AMPL (Fourer 1993), and then solved using standard software from OSL (IBM 1990).

### 3 COORDINATION: PUTTING THE PIECES BACK TOGETHER

Coordination strategies based on engineering intuition are usually posed in an ad hoc manner and cannot guarantee that they converge to the same solution set as that of the undecomposed problem. Rigorous coordination strategies tend to make strong assumptions, such as linearity or convexity of all models, which may be unenforceable.

In the next subsections we will describe two approaches that attempt to strike a compromise between rigor and practical value: decomposition synthesis and sequential decomposed programming. In the third subsection we will describe the hierarchical overlapping coordination method, which has some promising characteristics for distributed product development.

#### Decomposition synthesis

One may pose the “system” problem so that a particular partitioning formulation is facilitated (Krishnamachari and Papalambros 1997). This idea motivates a *decomposition synthesis* strategy, where a problem is first partitioned without specifying a system objective. Then a system objective is composed using criteria from each subsystem so that the system solution will be a Pareto optimum of the subsystem solutions.

The methodology proposed in (Krishnamachari and Papalambros 1997) mainly focuses on synthesizing optimal design problems (ODP) that can be solved by a primal hierarchical decomposition method (Wagner and Papalambros 1993). A block-angular structure is first identified for the general design problem (GDP), which lacks a design objective. An ODP is then created that can be hierarchically decomposed based on this structure. Formally, a GDP is first cast into the form

$$\begin{aligned}
 \mathbf{g}_0(\mathbf{x}_0) &\leq \mathbf{0} \\
 \mathbf{h}_0(\mathbf{x}_0) &= \mathbf{0} \\
 \mathbf{g}_i(\mathbf{x}_0, \mathbf{x}_i) &\leq \mathbf{0} \quad i= 1, \dots, K \\
 \mathbf{h}_i(\mathbf{x}_0, \mathbf{x}_i) &= \mathbf{0} \quad i= 1, \dots, K
 \end{aligned} \tag{5}$$

that has a master problem and  $K$  subproblems with a block-angular structure similar to that in Figure 1(b). This type of

structure can be identified using the problem partitioning techniques described in Section 2 above. A hierarchically decomposed ODP is then synthesized by composing a weighted additive objective selecting criteria  $f_0$  and  $f_i$  from constraints  $g_0$  and  $g_i$  in Eq. (5), as shown in Eq. (6).

$$\begin{aligned}
& \underset{\mathbf{x}}{\text{minimize}} && f_0(\mathbf{x}_0, \mathbf{w}_0) + \sum_{i=1}^K f_i(\mathbf{x}_0, \mathbf{x}_i, \mathbf{w}_i) \\
& \text{subject to} && \\
& \mathbf{g}_0(\mathbf{x}_0) \leq \mathbf{0} && \\
& \mathbf{h}_0(\mathbf{x}_0) = \mathbf{0} && \\
& \mathbf{g}_i(\mathbf{x}_0, \mathbf{x}_i) \leq \mathbf{0} && i = 1, \dots, K \\
& \mathbf{h}_i(\mathbf{x}_0, \mathbf{x}_i) = \mathbf{0} && i = 1, \dots, K
\end{aligned} \tag{6}$$

### Sequential decomposed programming

In another approach, existing nonlinear programming algorithms (NLP) are modified to take advantage of problem structure without losing their basic convergence properties. This can be particularly effective in sequential optimization methods, hence the term *sequential decomposed programming*. This approach has been successfully applied to sequential quadratic programming and trust region algorithms (Nelson 1997; Nelson and Papalambros 1998a).

Sequential decomposed programming accommodates the special type of structure known as hierarchically decomposable nonlinear programming and depicted in Eq. (6). Several (perhaps all) functions may depend on the vector of linking variables  $\mathbf{x}_0$ . Vectors  $\mathbf{x}_i$ ,  $i \neq 0$  are vectors of local variables since there are  $K$  sets of functions, identified with the index  $i$ ,  $\mathbf{g}_i(\mathbf{x}_0, \mathbf{x}_i)$ ,  $\mathbf{h}_i(\mathbf{x}_0, \mathbf{x}_i)$  and  $f_i(\mathbf{x}_0, \mathbf{x}_i, \mathbf{w}_i)$  that depend only on the linking variables  $\mathbf{x}_0$  and the respective local variables  $\mathbf{x}_i$ . If the linking variables are held constant, then Eq. (6) can be posed as  $K$  smaller and separate nonlinear programs, called subproblems and shown in Eq. (7), that can be solved independently.

$$\begin{aligned}
& \underset{\mathbf{x}_i}{\text{minimize}} && f_i(\mathbf{x}_0, \mathbf{x}_i, \mathbf{w}_i) \\
& \text{subject to} && \\
& \mathbf{g}_i(\mathbf{x}_0, \mathbf{x}_i) \leq \mathbf{0} && \\
& \mathbf{h}_i(\mathbf{x}_0, \mathbf{x}_i) = \mathbf{0} &&
\end{aligned} \tag{7}$$

In a typical hierarchical framework a master problem is solved in terms of  $\mathbf{x}_0$ . The predicted value for the optimal  $\mathbf{x}_0$  is passed to the subproblems (7), each of which is solved with respect to each  $\mathbf{x}_i$ . The optimal value of  $\mathbf{x}_i$  is returned to the master problem and the process is repeated until some convergence criterion is satisfied.

Nelson and Papalambros (1998a) indicate that care must be taken with the extra step in which the linking variables are held constant in order to retain the properties of the original algorithm. The step that in the original algorithm is usually called “the direction finding problem” or “the approximate problem” is referred to here as “the coordination problem,” because it represents a decision making process between the subproblems. To retain the global convergence properties of the original algorithm, coordination between subproblems is performed using an approximate problem (for example, a quadratic program) that has the same form as in the original algorithm. That is, coordination uses all of the constraints

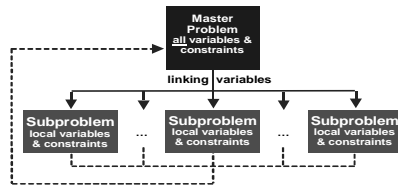


Figure 5. Sequential decomposed programming

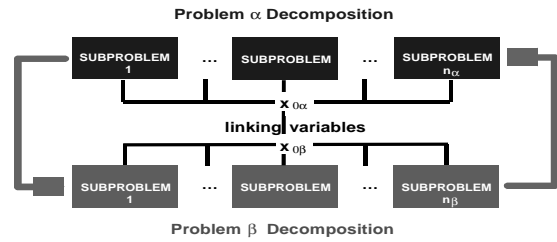


Figure 6. Hierarchical overlapping coordination

and all of the variables in the original problem, as depicted in Figure 5. In order to retain any established local convergence properties, the subproblems are not solved independently when near a solution, i.e., the additional step is not used. Finally, the subproblems are used not only to improve their respective objective functions while maintaining or obtaining feasibility, but also to give better estimates of other quantities used in the algorithm, such as penalty parameters, Hessian estimates, and trust region radii.

### Hierarchical overlapping coordination

Hierarchical overlapping coordination (HOC) (Macko and Haimes 1978) simultaneously uses two or more design problem decompositions, each of them associated with different partitions of the design variables and constraints. Michelena, et al. (1999) showed some extensions that can make the approach more applicable to design problems. A HOC strategy may reflect, for example, matrix-type organizations structured according to product lines or subsystems and the disciplines involved in the design process. Partitioning methods described in Section 2 can be used to generate the required problem decompositions. Coordination of the design subproblems is achieved by the exchange of information between decompositions.

Figure 6 illustrates how the HOC algorithm operates for two problem decompositions ( $\alpha$  and  $\beta$ ). Briefly, the algorithm can be described as follows:

- Step 1: Fix linking variables  $x_{0\alpha}$ , and solve Problem  $\alpha$  by solving  $p_\alpha$  independent subproblems, such as those in Eq. (7).
- Step 2: Fix linking variables  $x_{0\beta}$  to their values determined in Step 1, and solve Problem  $\beta$  by solving  $p_\beta$  independent subproblems.
- Step 3: Go to Step 1 with the fixed values of  $x_{0\alpha}$  determined in Step 2.
- Step 4: Repeat these steps until convergence is achieved.

The linking variables for one of the decompositions are fixed at values that result from the solution of a number of independent subproblems associated with the other decomposition. In general, the accumulation point achieved in Step 4 is not necessarily an optimal solution of the original problem. Convergence of the algorithm depends on the way the model decompositions interact with each other. A sufficient condition for convergence that can be computationally verified for decompositions of nonlinear convex problems was presented in Michelena et al. (1999). This condition together with model partitioning methods can help in generating appropriate problem decompositions.

In the next two sections we will describe two applications of decomposition strategies applied to product development: setting design targets and designing collections of products simultaneously.

## 4 DESIGN TARGET CASCADING

An important phase in product development of complex artifacts is an early determination of key design targets for the major components or subsystems of the artifact. The process usually starts with a statement of design targets (or mission specifications) dictated by market and other top level considerations, see Figure 7. These top level targets must be “cascaded” to the rest of the system so that (i) major parts of the system can be designed independently using their own local targets, and (ii) consistency of local targets to each other and to the system as a whole is maintained. Clearly a top-down hierarchical dictation of targets will not work without some iterative “rebalancing” that will guarantee feasibility. A formalism employing a partitioning and coordination strategy becomes eminently appealing.

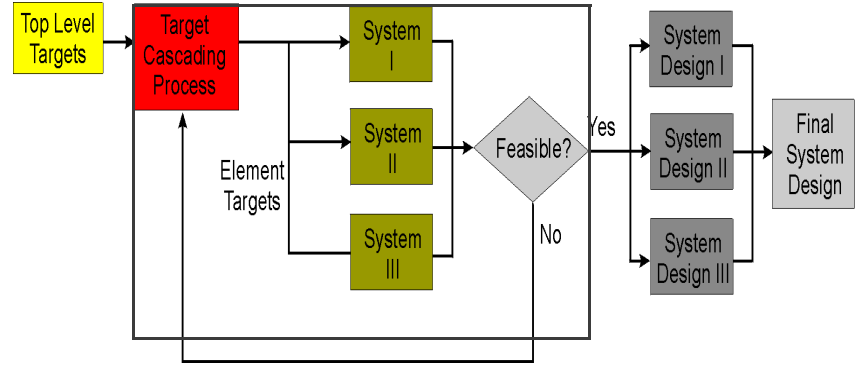


Figure 7. Target cascading as an element of a concurrent design process

A formal target cascading process can be stated as a mathematical optimization problem, assuming that appropriate models for the systems’ functions are available. These models, in general, must be simple enough requiring no detailed design information but powerful enough to capture important interactions among subsystems and components, whose performance needs may be competing. The original design problem can be stated as follows:

$$\begin{aligned}
 & \text{minimize } \|T - R\| \\
 & \quad \quad \quad x \\
 & \text{where } R = r(x) \\
 & \text{subject to} \\
 & g_i(x) \leq 0 \quad \quad \quad i = 1, \dots, m_i \\
 & h_j(x) = 0 \quad \quad \quad j = 1, \dots, m_e \\
 & x_k^{\min} \leq x_k \leq x_k^{\max} \quad \quad k = 1, \dots, n
 \end{aligned} \tag{8}$$

The objective is defined as the discrepancy between the target  $T$  and the response  $R$  obtained from the analysis model  $r(x)$ ;  $g$  and  $h$  are inequality and equality design constraint vectors, and the design variable  $x$  is defined within lower and upper bounds,  $x^{\min}$  and  $x^{\max}$ . Problem (8) can be reformulated as the multilevel problem shown in Figure 8. Since the “system level” is located in the middle of the overall hierarchy, this formulation is the most comprehensive, capturing all interactions, through linking variables, target responses from the lower level (superscript  $L$ ), and target responses from the upper level (superscript  $U$ ). At the system level, the problem can be stated mathematically as shown below in Eq. (9): minimize the deviations for system responses and subsystem linking variables, subject to system design constraints and tolerance constraints that coordinate subsystem responses and component design linking variables.

In the model below the objective function minimizes the discrepancy between current system level responses  $R_s$  and the targets set at the upper level  $R_s^U$ , as well as between subsystem linking variables  $y_{ss}$  and the targets set at the upper level  $y_{ss}^U$ . Therefore,  $R_s^U$  and  $y_{ss}^U$  are determined by solving an equivalent problem at the higher level. Target deviation tolerances are minimized to achieve consistent design with minimum discrepancies between the subsystem level responses  $R_{ss}$  and the target responses  $R_{ss}^L$  from the subsystem design problem, as well as between the component

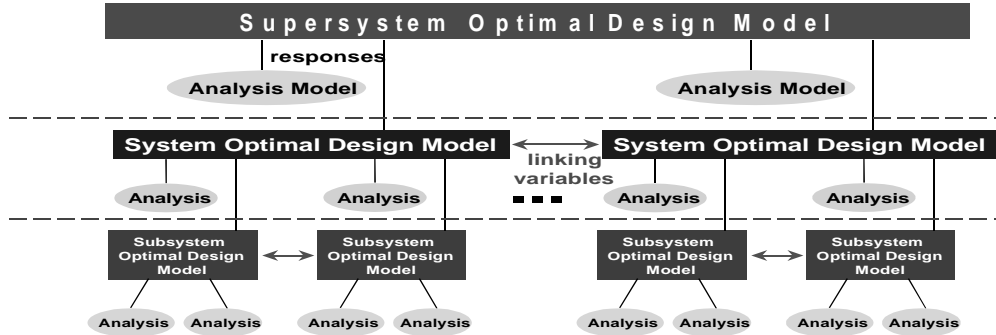


Figure 8. Multilevel representation of the target cascading problem (Kim et al. 2000)

level linking variables  $y_c$  and the target values  $y_c^L$  from the subsystem design problem.

$$\begin{aligned}
 & \text{minimize} \quad \left\| \mathbf{R}_s - \mathbf{R}_s^U \right\| + \left\| \mathbf{y}_{ss} - \mathbf{y}_{ss}^U \right\| + \varepsilon_R + \varepsilon_y \\
 & \mathbf{x}'_{ss}, \mathbf{y}_{ss}, \mathbf{y}_c, \mathbf{R}_{ss}, \varepsilon_R, \varepsilon_y \\
 & \text{where} \quad \mathbf{R}_s = \mathbf{r}_{s \leftarrow ss}(\mathbf{R}_{ss}, \mathbf{x}'_{ss}, \mathbf{y}_{ss}) \\
 & \text{subject to} \\
 & \left\| \mathbf{R}_{ss} - \mathbf{R}_{ss}^L \right\| \leq \varepsilon_R, \left\| \mathbf{y}_c - \mathbf{y}_c^L \right\| \leq \varepsilon_y, \varepsilon_R \geq 0, \varepsilon_y \geq 0 \\
 & \mathbf{g}_s(\mathbf{R}_{ss}, \mathbf{x}'_{ss}, \mathbf{y}_{ss}) \leq \mathbf{0} \\
 & \mathbf{h}_s(\mathbf{R}_{ss}, \mathbf{x}'_{ss}, \mathbf{y}_{ss}) = \mathbf{0} \\
 & \mathbf{x}_{ss}^{\min} \leq \mathbf{x}'_{ss} \leq \mathbf{x}_{ss}^{\max} \quad \mathbf{y}_{ss}^{\min} \leq \mathbf{y}_{ss} \leq \mathbf{y}_{ss}^{\max}
 \end{aligned} \tag{9}$$

.A proper coordination strategy is still needed to solve this problem. This is not a trivial task. The formulation here is reminiscent of the proposed “collaborative optimization” approach (Braun and Kroo 1997). As has been shown recently (Alexandrov and Lewis 2000) collaborative optimization uses a model formulation and coordination strategy that can violate constraint qualifications a priori, so that Karush-Kuhn-Tucker optimality conditions would not be applicable. This issue remains a current research challenge

The target cascading process has been applied successfully to some early studies in both academic and industrial settings (Michelena et al. 1999, Kim et al. 2000). The approach is currently studied as part of the vehicle development process in automotive new product development. In a practical industry implementation a key element of success is the availability of appropriate complexity models. Even for the same product, targets change with time possibly in a period of months, so the overall process must be dynamic. One must envision a multiphase cascading process where the above formalism is applied to models of increased complexity and detail, as the product gets a more definitive characterization. In a sense one must have a cascade of models of increasing complexity in addition to a cascade of models in the hierarchy shown earlier. The challenges from both an engineering and a management viewpoint are obvious, indicative of the difficulties faced when realistic complexity is being addressed.

## 5 DESIGNING PRODUCT COLLECTIONS

Designing a collection of products simultaneously, as opposed to a single product, is an idea that has a variety of appeals. One attraction is the desire to avoid commitment to a single design until the later phases of the product development process when commitments are made closer to production, hence presumably with less risk. Another attraction is the desire to limit costs by imposing some commonality, thereby deriving a family of variants from a single “platform.” Costs in manufacturing facilities and product development resources are reduced, and rapid exercise of product options to changing market needs is enhanced.

In general if two or more products share some parts the performance of the individual product will be likely compromised, compared to its performance without the commonality restriction. An optimization formulation can quantify this change in performance, and also compare the effect of sharing different sets of components. In a formal model, one must compare the best possible designs when sharing parts to the best possible designs when the products are not sharing parts. A single multicriteria optimization problem can capture the decision required in evaluating the entire platform (Nelson et al. 1999). Assuming that a criterion depends only on variables  $x_i$ , we pose the problem

$$\begin{aligned}
 &\text{minimize} && f_i(x_i) && i = 1 \dots p \\
 &&& g_i(x_i) \leq \mathbf{0} && i = 1 \dots p \\
 &&& h_i(x_i) = \mathbf{0} && i = 1 \dots p \\
 &\text{subject to} && x_{i, k_1} = x_{j, k_2} && (k_1, k_2) \in P_{ij} \\
 &&& && i, j = 1 \dots p \\
 &&& && i < j
 \end{aligned} \tag{10}$$

Let  $P_{ij}$  be a set of index pairs used to represent a set of equality constraints. If products  $i$  and  $j$  share some parts, then  $P_{ij}$  contains the index pairs of the design variables describing the common parts. Variants in a platform configuration is collectively defined by a distinct set of index pairs. To compare two different product platforms, solutions are compared with different sets of index pairs, say  $\{P_{ij}\}$  and  $\{Q_{ij}\}$ . For a specific set of  $P_{ij}$  the solutions of the problem above are a Pareto set. Let different superscripts represent optimal values from different platform configurations. For two products A and B in a platform, a superscript *circle* represents optimal quantities for the null platform ( $f_A^\circ$  and  $f_B^\circ$ ), i.e., independently designed products. A superscript *bullet* ( $f_A^\bullet$  and  $f_B^\bullet$ ) represents optimal quantities for the platform with the common parts. The individual minima  $f_i^\bullet$  of Eq. (10) are defined as the extreme values of the Pareto set, see Figure 9.

These are the solutions to Eq. (10) with only one of the scalar functions ( $f_A$  or  $f_B$ ) used as an objective. The utopia point ( $f_A^\bullet, f_B^\bullet$ ), is likely not as good as the point representing separate designs ( $f_A^\circ, f_B^\circ$ ). The additional equality constraints of the product platform imply that the feasible space is smaller so the designer of a product platform should *expect* to give up some acceptable amount of performance (Nelson et al. 1999).

In a design strategy each possible pair of products that can share parts is identified. The individual optima for each configuration is computed to decide whether a trade-off study should be pursued. If so, the relevant multicriteria problem of Eq. (10) is solved to calculate the Pareto set. The design that offers the best value for all appropriate products while still allowing for the benefits of having a flexible product platform is then chosen. This strategy has been applied to vehicle platform design (Fellini et al. 2000) as a first step towards a rigorous portfolio development strategy. The point here, however, is that the problem in Eq. (10) is generally difficult to solve as the size of the problem increases, and in practice a hierarchical decomposition strategy must be employed, similar to what was discussed in the previous sections.

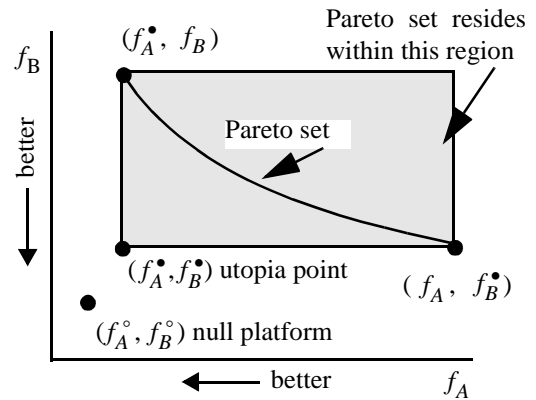


Figure 9. Pareto set for a platform with two products A and B sharing one or more components.

## 6 SYSTEM-LEVEL DESIGN AND CONTROL

We now turn our attention to “smart” artifacts, another example of system design where the ideas of the earlier sections have bearing. Design of a modern artifact is likely to include a control function that allows the artifact to adapt to changing environmental conditions. Traditionally, “design” and “control” have been treated as separate engineering functions. A little reflection reveals that designing the embodiment of an artifact and designing its controller are both design functions. Indeed, “control” means defining the control function and then designing the embodiment of a physical controller. A system-level design and control requires looking at these design activities in a simultaneous manner.

$$\text{minimize } f = a_1 d$$

variable:  $d$

coupling parameter:  $b$

simple parameters:  $\mathbf{a} = \{a_1, a_2, a_3\} \geq \mathbf{0}$

subject to:

$$a_2 b + a_3 - d \leq 0$$

output:  $v = d^*$

optimum:  $d^* = a_2 b + a_3$

### (a) Optimal design Problem

$$\begin{aligned} \text{minimize } J &= \int_0^{t_f} (x^T u_1 x + z^T u_2 z) dt \\ &= \frac{u_5^2 (u_1 + u_2 p^2)}{2(v - u_3 p)} [e^{2(v - u_3 p)t_f} - 1] \end{aligned}$$

variable:  $p$

coupling parameter:  $v$

simple parameters:  $\mathbf{u} = \{u_1, u_2, u_3, u_4, u_5, t_f\} \geq \mathbf{0}$

subject to:

$$\dot{x}(t) = vx + u_3 z = (v - u_3 p)x \Rightarrow x(t) = e^{(v - u_3 p)t}$$

$$y = u_4 x$$

$$z = -px$$

$$x(t = 0) = u_5$$

$$\text{output: } b = x(t = 1) = e^{(v - u_3 p^*)}$$

$$\text{optimum: } p^* = (u_1 v + \sqrt{(u_1^2 v^2 + u_1 u_2 u_3^2)}) / u_3^2$$

### (b) Optimal Gains Problem

$$\text{minimize } \phi = w_1 f + w_2 J$$

where:  $f = a_1 d$

$$\begin{aligned} \text{and } J &= \int_0^{t_f} (x^T u_1 x + z^T u_2 z) dt \\ &= \frac{u_5^2 (u_1 + u_2 p^2)}{2(v - u_3 p)} [e^{2(v - u_3 p)t_f} - 1] \end{aligned}$$

variables:  $d, p$

coupling quantities:  $b, v$

simple parameters:  $\mathbf{a} = \{a_1, a_2, a_3\} \geq \mathbf{0}$

$$\mathbf{u} = \{u_1, u_2, u_3, u_4, u_5, t_f\} \geq \mathbf{0}$$

subject to:

$$\dot{x}(t) = vx + u_3 z = (v - u_3 p)x \Rightarrow x(t) = e^{(v - u_3 p)t}$$

$$y = u_4 x$$

$$z = -px$$

$$x(t = 0) = u_5$$

$$l: -p \leq 0$$

$$g: a_2 b + a_3 - d \leq 0$$

$$\beta: e^{(v - u_3 p^*)} - b \leq 0$$

$$\gamma: d - v \leq 0$$

### (c) Combined Design and Control Problem

Figure 10. Model statements for the design, control and combined problem in the simplified example (Reyer and Papalambros, 2000)

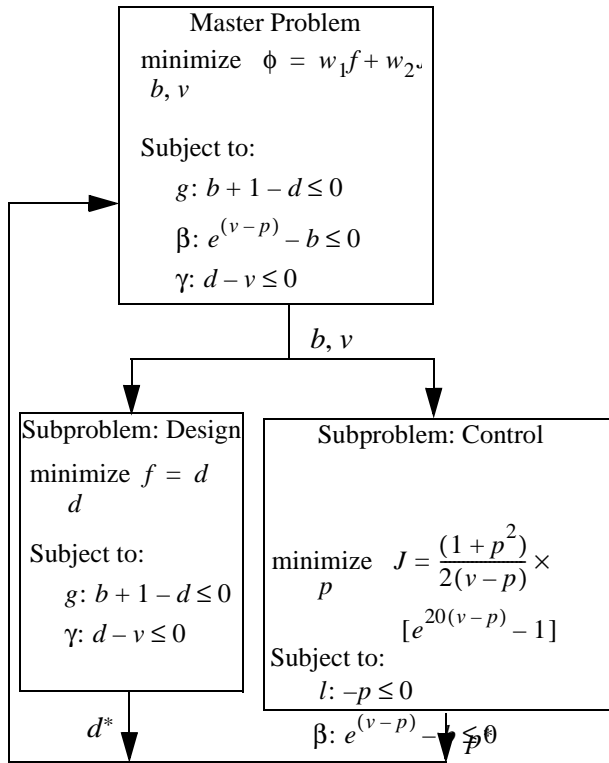


Figure 11. Partitioned concurrent strategy

The design problem can be defined as shown in Figure 10(a). The model has one design variable  $d$ , one input coupling parameter  $b$ , several simple parameters  $a$ , and one output coupling parameter  $v$ . The problem is linear with a single unique global optimal solution which is constraint bound (i.e., fully determined by active constraints). The control problem is an optimal gains one with finite-dimensional variables as shown in Figure 10(b). The model has one control variable  $p$ , one input coupling parameter  $v$ , several simple parameters  $u$ , and an output coupling parameter  $b$ . For this simple problem the control strategy is equivalent to both a proportional feedback and an LQR controller. The control optimum is based on the LQR solution and has an analytical form. The system-level problem is defined with an objective that is the weighted sum of the design and control objectives, representing the Pareto solution of the two problems, Figure 10(c). The combined model has two variables  $d$  and  $p$ , two coupling quantities  $b$  and  $v$ , two sets of simple parameters  $a$  and  $u$ . The model includes four inequality constraints: one from the control side  $l$ ; one from the design side  $g$ , and two for coupling  $\beta$  and  $\gamma$ . The coupling constraints are *directed* (Papalambros and Wilde 1988, 2000)—a critical issue in understanding how the different strategies work with more details found in the cited reference. The decomposition strategy ideas explored earlier can be used here to put intuitive ideas on a rigorous foundation. The partitioned problem has two linking variables, the coupling variables  $b$  and  $v$ , and two subproblems, one for design and one for control, Figure 11 and Figure 12. In the hierarchical representation the master problem has two variables,  $b$  and  $v$ , three constraints,  $g$ ,  $\beta$  and  $\gamma$ , and the weighted combined objective. The design subproblem has one variable  $d$ , two constraints,  $g$

A sequential design approach is often natural, as each task may be itself difficult and coupling them together could make the design task intractable. Problems with nonlinearities in the control tasks have particular difficulties. Decision criteria for each design task are often different and competing. Nevertheless, the initial design of the artifact affects its control characteristics, so if adaptive performance is important or expensive, control-related design criteria must be incorporated early in the design process. As products become increasingly complex, examining the tight coupling between embodiment design and control design becomes a critical step in the design process.

Designing a controller is generally a difficult *configuration design* problem. If a given configuration is assumed, an *Optimal Gain Problem* can be formulated that is time independent. In this context, the problem is broached in Reyer and Papalambros (1999). Here we can illustrate the main ideas using a simplified example from Reyer and Papalambros (2000). The term “simple” below means that the quantity in question is local to the problem (design or control), while the term “coupling” means that the relevant quantity appears in the other problem as well.

|          | $b$ | $d$ | $p$ | $v$ |
|----------|-----|-----|-----|-----|
| $f$      |     | ■   |     |     |
| $J$      |     |     | ■   | ■   |
| $g$      | ■   | ■   |     |     |
| $l$      |     |     | ■   |     |
| $\beta$  | ■   |     | ■   | ■   |
| $\gamma$ |     | ■   |     | ■   |

|     | $b$ | $v$ | $d$ | $p$ |
|-----|-----|-----|-----|-----|
| $f$ |     |     | ■   |     |
| $J$ |     |     | ■   | ■   |
| $g$ | ■   |     | ■   |     |
| $g$ |     |     | ■   | ■   |
| $b$ | ■   | ■   |     | ■   |
| $l$ |     |     |     | ■   |

Figure 12. Functional dependency tables for non-partitioned and partitioned problem

and  $\gamma$ , and the design objective  $f$ . Note that in the design subproblem, the control part of the system objective  $J$ , and the constraints  $l$  and  $\beta$ , are eliminated, since the design variable cannot affect them. Similarly, the control subproblem has one variable  $p$ , two constraints  $l$  and  $\beta$ , and the control objective  $J$ .

This partitioned problem must be solved with an appropriate coordination strategy. As explored in the cited reference, traditional design-control methods in general will not find the system optimum, either because they use no strategy equivalent to coordination or because the partition and the strategy employed miss elements of the combined problem shown above.

Even this simple example appears rather devious when addressing it from a system perspective. Problems of realistic complexity quickly become challenging. A deeper study of proper partitioning and coordination methods derived from the optimality conditions of the combined problem would be illuminating and hopefully lead to practical but rigorous system solution strategies.

## 7 CONCLUDING REMARKS

Partitioning can be now performed rigorously using the techniques described here. However, the optimal partitioning problem must be formulated having some intuition about the specific problem. Many different optimal partitions can be derived, some of which may be meaningful and some not. Therefore, the designer must still exercise judgment. In coordination the challenge is to prove “convergence” of the strategy. Convergence formally would mean that the non-partitioned problem and the partitioned one solved with coordination have the same solution set. This is often hard or impossible to prove for practical nonlinear problems. Alternatively, a problem may be defined in a meaningful *partitioned* form directly, as in the case of decomposition synthesis described above, where the system solution is by definition a Pareto solution of the subsystem problems. As complexity increases, the ideas described above can and should be extended to multilevel problems. The main extra difficulty there is controlling the computational cost.

Another challenge, occasionally insurmountable, is the presence of simulations in the models, namely, large, expensive implicit models, whose mathematical behavior as function generators is generally unknown, and frequently noisy. Dealing with noisy, expensive and likely non-convex functions is challenging, and particularly so for decomposition methods. Some discussion on this issue can be found in Sasena et al (2000) and Fellini et al (2000).

Finally, one should readily confess that decomposition strategies as described in the present context almost always incur additional computational cost, when compared with solving the undecomposed problem. Why then should one consider them at all? First, these methods may be the only hope to solve a problem whose complexity defies the alternative. But since coordination strategies may be ad hoc, even then a solution may be not known with certainty. The second reason for using a decomposition approach is simply to break down the problem to subproblems of sufficient size that a relatively experienced team of designers will have enough intuition to sanction the numerical results. From an optimization perspective, of course, the challenge is to develop such methods that are ever more robust and address an ever broader range of design problems.

## ACKNOWLEDGEMENTS

This review article was composed based on joint work with a number of collaborators at the University of Michigan, including R. Fellini, H.M. Kim, R.P. Krishnamachari, S. Nelson, M. Parkinson, M. Sasena, and J. Reyer. The work described has been supported by a variety of sponsors, including the Automotive Research Center at the University of Michigan under the auspices of the US Army TACOM, Ford Motor Co, General Motors Corp., and the US Department of Energy. The authors gratefully acknowledge the support of the sponsors and the contributions of our colleagues.

## REFERENCES

- Alexandrov, N., Dennis, Jr., J. E., Lewis, R. M., and V. Torczon, 1998. “A Trust Region Framework for Managing the Use of Approximate Models in Optimization,” *Structural Optimization*, Vol. 15, no. 1, pp. 16–23.
- Braun, R. D. and Kroo, I. M., 1997, “Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment,” in *Multidisciplinary Design Optimization: State-of-the-Art*, N. Alexandrov and M. Hussaini (eds.), SIAM, pp. 98–116

- Fellini, R., Papalambros, P.Y., and T. Weber, "A Decomposition Strategy for Product Platform Design with Application to Automotive Powertrains," *AIAA/USAF/NASA/ISSMO Symp. on Multidisc. Anal. and Optimization*, Long Beach, CA, Sept. 2000.
- Fourer, R., Gay, D., and Kernighan, B., 1993. *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, South San Francisco.
- IBM, 1990. *Optimization Subroutine Library Guide and Reference*, IBM Corporation, N.Y.
- Kim, H.M., Michelena, N. F., Papalambros, P. Y., and T. Jiang, 2000. "Target Cascading in Optimal System Design," *ASME Design Automation Conference*, Baltimore.
- Krishnamachari, R., and P.Y. Papalambros, 1997. "Optimal Hierarchical Decomposition Synthesis Using Integer Programming," *ASME J. of Mech. Design*, Vol.119, No.4, pp.440–447.
- Macko, D., and Y. Haimes, 1978. "Overlapping Coordination of Hierarchical Structures," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 10, pp.745-751.
- Michelena, N., and P.Y. Papalambros, 1995. "A Network Reliability Approach to Optimal Decomposition of Design Problems," *ASME J. of Mech. Design*, Vol. 117, No. 3, pp. 433–440.
- Michelena, N., and P.Y. Papalambros, 1997. "A Hypergraph Framework to Optimal Model-Based Decomposition of Design Problems," *Comp. Optim. and Appl.*, Vol. 8, No. 2, pp. 173–196.
- Michelena, N., Papalambros, P., Park, H.A., and Kulkarni, D., 1999. "Hierarchical Overlapping Coordination for Large-Scale Optimization by Decomposition," *AIAA Journal*, Vol. 37, No. 7, 1999, pp. 890–896.
- Michelena, N., Kim, H. M., and P. Y. Papalambros, 1999. "A System Partitioning and Optimization Approach to Target Cascading," *Proc. 12th Int. Conf. on Engineering Design*, (U. Lindemann et al., Eds.), Vol. 2, pp 1109–1112, Munich.
- Nelson, S.A., and P.Y. Papalambros, 1998a. "A Modified Trust Region Algorithm for Hierarchical NLP," *Structural Optimization*, Vol.16, 1998, pp. 19–28.
- Nelson, S.A., and P.Y. Papalambros, 1998b. "Exploiting Discrepancies in Function Computation Time Within Optimization Models," *ASME Design Automation Conference*, Atlanta. To appear as "The Use of Trust Region Algorithms to Exploiting Discrepancies in Function Computation Time Within Optimization Models," *ASME J. of Mech. Design*.
- Nelson, S.A., Parkinson, M.B., and P.Y. Papalambros, 1999. "Multicriteria Optimization in Product Platform Design," *ASME Design Automation Conference*, Las Vegas.
- Papadimitriou, C., and Steiglitz, K., 1982. *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
- Papalambros, P.Y., and D. J. Wilde, 1988, 2000. *Principles of Optimal Design: Modeling and Computation*. 1st Ed., 2d Ed.; Cambridge University Press, New York.
- Reyer, J.A., and P.Y. Papalambros, 1999. "Optimal Design and Control of an Electric DC Motor," *ASME Design Automation Conference*, Las Vegas.
- Reyer, J. A., and P. Y. Papalambros, 2000. "An Investigation into Modeling and Solution Strategies for Optimal Design and Control," *ASME Design Automation Conference*, Baltimore.
- Sasena, M.J., Papalambros, P.Y., and P. Goovaerts, "Metamodeling Sampling Criteria in a Global Optimization Framework," *AIAA/USAF/NASA/ISSMO Symp. on Multidisc. Anal. and Optimization*, Long Beach, CA, Sept. 2000.
- Wagner, T.C. and Papalambros, P.Y., 1993. "A General Framework for Decomposition Analysis in Optimal Design," *ASME Design Automation Conference*, DE–Vol. 65–2, New York, pp. 315–25.