A Network Reliability Approach to Optimal Decomposition of Design Problems

Methods for solving partitioned mathematical programming problems require that an appropriate structure suitable for decomposition be identified. This first step consists of identifying linking variables that effect independent subproblems coordinated by a master problem. This article presents a network reliability-based solution of the optimal decomposition problem that avoids subjective criteria to identify linking variables and partitions. The relationships among design variables are modeled as the processing units of a network. The design variables themselves are modeled as the communication links between these units. The optimal decomposition is attained by minimizing the network reliability while maximizing the number of operating links.

Introduction

The operations research community has studied the decomposition problem to improve computational efficiency and robustness in solving structured, partitioned optimization problems. However, identification of the partitioned problem form has remained a largely ad hoc task. In the design community, partitioning of design problems has received considerable interest for the purpose of improving coordination and information transfer across multiple disciplines (Sobieski, 1988, 1990; Bloebaum, 1991) and for streamlining the design process by adequate arrangement of the multiple design activities and tasks (see references below). Decomposition in the solution of large-scale system design problems allows for a conceptual simplification of the system, reduction in the dimensionality of the problem, more efficient computational procedures, different solution techniques for individual subproblems, simultaneous design, modularity, multiojective analysis, and efficient communication and coordination among the diverse groups involved in the design process.

This article presents a formal solution of the optimal model-based partitioning (OMBP) problem that does not resort to heuristics and subjective criteria for the generation of partitions. The solution of the OMBP problem proceeds and is at a higher level than, the solution of the underlying design problem. The design problem could be an optimal design problem (ODP) or a general design problem (GDP), depending on whether or not the formulation contains an objective function.

Decomposition strategies in the design literature are commonly classified as object (by physical components), aspect (by domains of knowledge), and sequential (by directed flow of elements or information) decomposition.

Steward (1981a, b), Rogers (1989), Eppinger et al. (1992, 1994), and Kusiak and Wang (1993a, b) applied sequential partitioning to the design sequence. Directed graphs and matrices are used to represent precedence relationships among the design tasks. These approaches identify groups of design tasks that can be ordered in a feed-forward sequence by detecting circuits among the interdependencies of the tasks. Steward used matrix transformations to minimize design iterations. Rogers used a rule-based system to generate a triangular form of the design structure matrix. Eppinger’s work is based on Steward’s matrix reordering, but it also includes subjective quantifiers of the strength of the dependencies among tasks. Kusiak and Wang proposed triangularization and diagonalization algorithms for the design structure and incidence matrices, respectively. They also proposed a branch and bound algorithm to identify overlapping design tasks or variables whose removal makes the design incidence matrix decomposable. The need for a precise definition of an input-output relation for each design task may limit the application of these techniques in situations where causality between design tasks is nonexistent or ill-defined. Also, these authors use heuristics or users’ input to identify “tears” of dependence relations between tasks (and, therefore, of circuits) whenever the structure of the problem is not sequentially decomposable.

Wagner and Papalambros (1993a, b) used an undirected graph representation of a design problem to partition it into subproblems. Their methodology, termed “Decomposition Analysis,” requires identification of “linking” (or “coordinating”) variables y that bring about independent design subproblems. These independent subproblems are associated with connected components in the graph representation of the problem after linking variables are deleted. The remaining variables in each connected subgraph are the “local” variables x_k of the corresponding subproblem k. Candidates for linking variables are recognized based on the number of design relations containing the candidate variables, the monotonicity or linearity of the variables in the relations, and an upper bound on the number of linking variables. Metrics used to test acceptable partitions include the total number of disjoint partitions and the sizes and relative sizes of the disjoint partitions.

Figure 1 shows a generic coordination strategy used by optimization methods for hierarchically partitioned problems. The master problem is solved for the linking variables y^* which are then input as parameters to the individual subproblems (solid arrows). Information on the dependence of the local variables with respect to the linking variables x_k(y^*) is fed back to the master problem (dashed arrows). The value of the overall objective function may provide a stopping criterion for this iterative process. The number of linking variables determines the size of the master problem, whereas the number and size of the disjoint partitions determine the number and size of the subproblems, respectively. There is a vast number of publications on solving partitioned problems in both the design optimization and the mathematical programming literature. A recent review can be found in Wagner’s dissertation (Wagner, 1993).

Although the decomposition scheme shown in Fig. 1 depicts a hierarchical problem structure, a nonhierarchical problem decomposition easily follows if the linking variables are identified as the communication links between subproblems. In this case,
bidirectional information flow between subproblems exists, so the information coupling is not hierarchical. A nonhierarchical coordination strategy, such as those proposed by Pan and Díaz (1990), Sobieski (1988, 1990), Wu and Azarm (1992), or Bloebaum (1991), is needed to approximate the effect of one subproblem on another. The main difference between a master problem and a nonhierarchical coordination strategy is that the coordinating variables in nonhierarchical methods are not necessarily original problem variables but represent mechanisms to account for subproblem coupling.

The OMBP problem will be formulated as a multiplicative optimization problem with two conflicting objectives, namely:

- Minimize size of master problem by minimizing number of linking variables, and
- Minimize size of subproblems by maximizing number of partitions, these being partitions of similar size.

Once the OMBP problem is solved, i.e., an optimal set of linking variables and associated partitions has been identified, a decomposition method from the class shown in Fig. 1 may be used to solve the design problem. As explained later in this article, the above objectives can be extended to account for the strength of function dependence on variables by assigning weights to the design variables.

**Representation of a Design Problem**

The following classical forms of the GDP and ODP are assumed in this article:

**General Design Problem**

\[
\begin{align*}
& \text{find } x \in \mathbb{R}^n \\
& \text{such that } h(x) = 0 \\
& g(x) \leq 0 \\
& \text{and } f(x) \text{ is minimized}
\end{align*}
\]

**Optimal Design Problem**

\[
\begin{align*}
& \text{find } x \in \mathbb{R}^n \\
& \text{such that } h(x) = 0 \\
& g(x) \leq 0 \\
& \text{and } f(x) \text{ is minimized}
\end{align*}
\]

Subject to:

\[
\begin{align*}
& f_1 = 0.4x_1^{0.67}x_2^{0.67} \\
& f_2 = 0.4x_2^{0.67}x_3^{0.67} \\
& f_3 = 10 - x_1 \\
& f_4 = -x_2 \\
& g_1 = 0.1x_1 + 0.588x_3x_7 - 1.0 \leq 0 \\
& g_2 = 0.1x_1 + 0.1x_3 + 0.588x_5x_8 - 1.0 \leq 0 \\
& g_3 = 4x_3x_7^{0.1} + 3x_7^{0.1}x_7^{0.1} + 0.588x_3^{1.3}x_7 - 1.0 \leq 0 \\
& g_4 = 4x_4x_6^{0.1} + 2x_4^{0.1}x_6^{0.1} + 0.588x_4^{1.3}x_6 - 1.0 \leq 0
\end{align*}
\]

Figure 2(a) shows the corresponding FDT. A shaded box indicates a "True" Boolean value. Figure 2(b) shows the FDT for the same problem after \( x_1 \) has been selected as linking variable and rows and columns have been reordered to reveal two

**Nomenclature**

- \( (0, e) = (e, \ldots, e, 0, e, \ldots, e) \)
- \( (1, e) = (e, \ldots, e, 1, e, \ldots, e) \)
- \( E = \text{vector of state random variables: } e_i \text{ for each } i \text{th edge} \)
- \( e = \text{vector of binary state variables: } e_i \text{ for each } i \text{th edge} \)
- \( E(G) = \text{set of edges in graph } G \)
- \( E[X] = \text{expected value of random variable } X \)
- \( f = \text{objective function} \)
- \( \phi_{at} = \text{all-terminal network state} \)
- \( \phi_{nt} = \text{network resilience} \)
- \( G = \text{graph} \)
- \( g, h = \text{vectors of constraint functions} \)
- \( h_1, g_1 \)
- \( G-i = \text{graph obtained from graph } G \)
- \( G-i = \text{graph obtained from graph } G \)
- \( G-i = \text{after deleting } i \text{th edge} \)
- \( G-i = \text{after contracting } i \text{th edge} \)
- \( m = \text{number of design variables or (hyper-)edges} \)
- \( n = \text{number of design functions or nodes} \)
- \( P[X] = \text{probability of event } [X] \)
- \( R_{\text{at}} = \text{all-terminal network reliability} \)
- \( R_{\text{pc}} = \text{pair-connected network reliability} \)
- \( V(G) = \text{set of nodes in graph } G \)
- \( w = \text{weighting coefficient} \)
- \( x = \text{vector of design variables } x_i \)
- \( \mathbb{R}^n = \text{n-dimensional Euclidean space} \)
- \( \lor = \text{operator defined by } (e_i \lor e_j) = 1 - (1 - e_i)(1 - e_j) \)
- \( \langle f \rangle = \text{Lovász extension of } f \)
partitions of the problem: one containing functions \( \{ f_1, f_4, g_2, g_3 \} \) and local variables \( \{ x_2, x_4, x_5, x_6 \} \), and the other containing functions \( \{ f_1, g_1, g_2 \} \) and local variables \( \{ x_3, x_5, x_7 \} \).

A design problem (and therefore its FDT) may also be represented by an undirected, linear graph. This type of representation does not require a priori knowledge of input-output relations or causality between variables. Wagner and Papalambros (1993a) used a graph representation that assigned a clique to each variable. They 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. We will instead use a tree to connect functions dependent on the same variable, with the significant reduction in the number of edges in the representation. An edge is labeled with the variable name the functions associated with the edge’s incident vertices depend on. Figure 3(a) shows trees associated with variables \( x_1 \) and \( x_2 \) in the FDT of Fig. 2. Figure 3(b) shows the graph representation of problem of Eq. (1). This graph is constructed by joining the trees corresponding to variables.

To minimize the number of edges in the graph representation, the function with the greatest row count is chosen as root of the individual trees associated to each variable. In Fig. 3(a), \( g_2 \) is selected as root node of the trees for \( x_1 \) and \( x_2 \) since it has the greatest row count (\( =4 \)) in the FDT shown in Fig. 2. Also, parallel edges are replaced by a single edge with appropriate label, as shown in Fig. 3(b).

**Motivation for Network Reliability Formulation of OMBP Problems**

The above undirected graph representation of a design problem is motivation for formulating the optimal decomposition problem as a network reliability optimization problem. Recalling that we want to minimize the number of linking variables as well as to maximize the number of partitions, we can formulate the OMBP problem as a network optimization problem with two conflicting objectives: (1) maximize the number of functioning links, and (2) minimize a measure of overall network reliability. Functioning links are identified with local variables, while failed links are identified with linking variables. On the other hand, common sense tells us that the more connected a network is, the more reliable it is.

This network optimization paradigm for the optimal decomposition problem may also be thought as one of selecting critical communication links, the linking variables, and assigning their control to a top decision-maker, the master problem. The control of 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, and they are identified by finding the Pareto points shown in Fig. 4. 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.

A problem containing \( m \) variables can be optimally partitioned in at most \( m \) different ways, each decomposition corresponding to \( 0, 1, 2, \ldots, \) or \( (m-1) \) linking variables. The selection of the optimal decomposition depends on the trade-off between the size of the master problem and the size of the subproblems. In general, the cost of execution of the master problem is increasing with the number of linking variables, whereas the cost of execution of the subproblems solved in parallel is decreasing with both the number of linking variables and the number of disjoint partitions. These measures also depend on the solution methods and characteristics of the functions involved in the master problem and subproblems. The Pareto solution of an OMBP problem (the aim of this work) results

---

1. In a linear graph an edge connects two vertices. A hypergraph, in which a link may connect more than two vertices, would be a more natural representation of a FDT. However, same labeling for two or more edges will make up for the limitations of linear graphs. See forthcoming article in this journal and by the same authors on a hypergraph-based formulation of the OMBP problem.

2. A \( k \)-clique is a graph with \( k \) vertices where each vertex is connected to every other vertex in the graph. A \( k \)-clique has \( \binom{k}{2} \) edges.

3. A tree is a connected graph without any circuits. A tree containing \( k \) vertices has \( (k-1) \) edges. A graph is connected if a path exists from every vertex to every other vertex.

---

**Figure 4** Pareto solution of an OMBP problem
in a relation between the Pareto optimal number of linking variables and the number of partitions that may be used to estimate the optimal number of linking variables.

Additional motivation for a network reliability formulation is that the actual system design process corresponds to an organizational structure (of software and/or people) and an associated management (and, therefore, communication) process. Decomposition and coordination results obtained from our formulation can be directly implemented in an organization to reduce product development time.

All-Terminal Network Reliability

A communication network enables transportation of information between end users, such as terminals, devices, computer systems, and the like. Failures in a network may arise in a number of different ways, from failures caused by control software to failures at the topological level, such as component wearout. A network may be modeled as a connected undirected graph \( G = (V(G), E(G)) \) consisting of a set \( V(G) \) of \( n \) nodes or vertices representing communication centers, and a collection \( E(G) \) of \( m \) undirected edges representing bidirectional communication links. It is assumed that nodes do not fail, and hence all network failures are consequences of edge failures.

At a deterministic level, we distinguish between two states for both a network and its edges: a functioning state and a failed state. We assign a binary indicator variable \( e_i \) to edge \( i \), such that \( e_i = 1 \) if edge \( i \) is functioning, and \( e_i = 0 \) if edge \( i \) is failed.

At a probabilistic level, we assume that the operation of an edge is statistically independent from other edges’ performance, and suppose that the state variable \( E_i \) of the \( i \)-th edge is random with \( P[E_i = 1] = E[E_i] \). We refer to \( p_i \), the probability that edge \( i \) functions, as the reliability of edge \( i \).

The all-terminal network state \( \Phi_\alpha \) is defined such that \( \Phi_\alpha = 1 \) if there is a path between every pair of nodes, and \( \Phi_\alpha = 0 \) otherwise. That is, \( \Phi_\alpha \) is one if the graph contains at least a functioning spanning tree. Similarly, the all-terminal network reliability \( R_\alpha \) is defined as the probability that for every pair \( u_1, u_2 \) of nodes there is a path between \( u_1 \) and \( u_2 \).

We assume that the all-terminal network state \( \Phi_\alpha \) is determined completely by the state of the edges \( e = (e_1, e_2, \ldots, e_m) \), so we may write \( \Phi_\alpha = \Phi_\alpha(e) \). Likewise, the all-terminal network reliability is given by \( R_\alpha = P[\Phi_\alpha(E) = 1] = E[\Phi_\alpha(E)] \), where \( E = (E_1, E_2, \ldots, E_m) \). Under the assumption of statistical independence we may represent the network reliability as \( R_\alpha = R_\alpha(p) \), where \( p = (p_1, p_2, \ldots, p_m) \).

The all-terminal network state of a tree network is one if all the edges are functioning, otherwise it is zero; that is, \( \Phi_\alpha(e) = e_1 \cdot e_2 \cdot \ldots \cdot e_m \). Similarly, \( R_\alpha(p) = p_1 \cdot p_2 \cdot \ldots \cdot p_m \). The all-terminal network state of a network with two nodes and \( m \) edges in parallel connecting these nodes is one if at least one edge is functioning, otherwise it is zero; that is, \( \Phi_\alpha(e) = 1 - (1 - e_1)(1 - e_2) \cdot \ldots \cdot (1 - e_m) \). Similarly, \( R_\alpha(p) = 1 - (1 - p_1)(1 - p_2) \cdot \ldots \cdot (1 - p_m) \).

Note that \( \Phi_\alpha(e) \) is a multilinear function. Moreover, \( \Phi_\alpha(e) \) is increasing in each argument since improving the performance of an edge cannot deteriorate the network reliability. This implies that \( \Phi_\alpha(0) = 0 \) and \( \Phi_\alpha(1) = 1 \). The following Lemma is a direct result of the multilinearity of \( \Phi_\alpha(e) \).

Lemma 1
The following expression holds for an all-terminal network state function:

\[ \Phi_\alpha(e) = e_\alpha \Phi_\alpha(1, e) + (1 - e_\alpha) \Phi_\alpha(0, e) \quad \text{for all} \quad e \quad (2) \]

Since the performance of the edges is independent and \( \Phi_\alpha(e) \) is a multilinear function, \( R_\alpha(p) \) is obtained from \( \Phi_\alpha(e) \) by replacing \( e_i \) by \( p_i \). From Lemma 1 and the independence of the edges follow the pivotal decomposition of the all-terminal network reliability function.

Lemma 2
The following expression holds for an all-terminal network reliability function:

\[ R_\alpha(p) = p_1 R_\alpha(1, p) + (1 - p_1) R_\alpha(0, p) \quad \text{for all} \quad p \quad (3) \]

The following expression for the derivative of the reliability function can be obtained from Lemma 2:

\[ \frac{\partial R_\alpha}{\partial p_1} = R_\alpha(1, p) - R_\alpha(0, p) \quad (4) \]

Pivotal decomposition formulas, such as those in Eqs. (2) and (3), combined with reduction of parallel edges can be used to compute a network state or reliability function. Satyanarayana and Chang (1983) have shown that for linear graphs an algorithm based on pivotal decomposition and parallel reductions produces a computational tree having the fewest leaves. In fact, the number of leaves in the computational tree is precisely the domination\(^3 \) \( D(G) \) of graph \( G \). The domination is smaller than the number of spanning trees, which could be exponential in the size of the graph. (A complete graph on \( n \) nodes has \( n^{n-2} \) spanning trees.) A pivotal decomposition algorithm can be executed in \( O(mD(G)) \) time and \( O((m - n + 1)|G|) \) space; however, when this type of algorithm is applied to networks with identical labels \( e_i \) for two or more edges, pivoting about an edge may entail the removal of more than one edge and node.

Network Resilience

All-terminal network state is a measure of connectivity of a graph since its value is one for connected graphs and zero for disconnected graphs. However, a disconnected graph may consist of components of very different sizes. Another measure of network connectivity that allows identifying partitions of similar size is valuable.

Resilience denotes the number of pairs of vertices that are connected in a network. At the probabilistic level, pair-connected reliability is the expected number of pairs of vertices that can communicate. Network resilience discriminates between network states with different failed edges, as shown in Fig. 5. The all-terminal network state for \( G_1 \) or \( G_2 \) is zero since these graphs are disconnected. However, the resilience for \( G_1 \) is \( \binom{5}{2} = 15 \) and for \( G_2 \) is \( \binom{5}{2} + \binom{2}{2} = 9 \). Thus, a network with a low resilience is likely to be partitioned in pieces of similar size.

\(^3\)A spanning tree of a connected graph contains all the vertices of the graph.
Colburn (1987) developed an algorithm for the resilience of series-parallel networks. Amin et al. (1987) and Siegrist et al. (1993) mainly focused on determining optimal tree and unicyclic graphs for pair-connected reliability. We present below the pivotal decomposition algorithm PIVOT to compute both the resilience and the all-terminal state of a general network. $\Phi_{PC}(e)$ and $R_{PC}(p)$ denote resilience and pair-connected reliability, respectively, which are also increasing and multilinear functions.

**Proposition 1**

The following identities hold for pair-connected reliability and resilience functions:

\[ R_{PC}(p) = p R_{PC}(1, p) + (1 - p) R_{PC}(0, p) \quad \text{for all } p \]  
(5)

\[ \Phi_{PC}(e) = e \Phi_{PC}(1, e) + (1 - e) \Phi_{PC}(0, e) \quad \text{for all } e \]  
(6)

**Proof:** Let $[E_i = 1]$ denote the event that edge $i$ is functioning and $[-E_i = 1]$ denote the event that edge $i$ is failed. Also, let $X$ be the random number of pairs of vertices that can communicate in a network. Since $E[X] = E[E[X|Y]]$ we have

\[ E[X] = P[E_i = 1] E[X | [-E_i = 1]] + P[-E_i = 1] E[X | [-E_i = 1]] \]

Equation (5) follows from this expression. Equation (6) follows since $R_{PC}(p) \rightarrow \Phi_{PC}(e)$ as $p \rightarrow e \in \{0, 1\}$.

Equations (5) and (6) may be used to compute the resilience and pair-connected reliability of a general network by computing these measures for tree structures. An expression like Eq. (4) may be used to compute the derivative of the pair-connected reliability function.

The pair-connected reliability of a graph may be evaluated in terms of two-terminal reliabilities $R_{uv}$—the probability that vertices $u$ and $v$ are connected in the graph—as follows:

\[ R_{PC} = \sum_{u \neq v} R_{uv} \]
(7)

where the sum is taken over all unordered pairs of distinct vertices in the graph. For a tree, there exists a unique path between every pair of vertices. Therefore, for any pair $u, v$ of vertices in a tree, the probability that $u$ and $v$ are connected equals the product of probabilities $p_i$ of the edges in the path between $u$ and $v$. In particular, if two vertices have been contracted to a single node, the associated two-terminal reliability is one; if the vertices belong to disconnected components, the two-terminal reliability is zero.

The following pivotal decomposition algorithms may be used to compute either the all-terminal state (with $j = 1$) or the resilience (with $j = 2$) of a network.

**PIVOT($G, j$):**

apply parallel reductions until no reduction applies

1. if $G$ is a tree and $j = 1$ then return the product of the indicator variables, $e_i$, of the edges in $G$

2. else if $G$ is a tree and $j = 2$ then return the resilience of the tree according to Eq. (7)

3. else if $G$ is a disconnected graph and $j = 1$ then return $0$

4. else choose the edge $i$ in $G$ whose associated variables have the smallest cumulative column count in the FDT and return $e_i$PIVOT($G \setminus i, j$) + $(1 - e_i)$PIVOT($G \setminus i, j$)

$G \setminus i$ is the graph obtained from $G$ by contracting edge(s) with indicator $e_i$, i.e., the nodes adjacent to edge(s) with indicator $e_i$ are collapsed into a single node and the edge(s) removed. $G \setminus i$ is the graph obtained from $G$ by deleting edge(s) with indicator $e_i$. Edge removal is equivalent to column deletion in the FDT, whereas edge contraction is equivalent to column deletion and logical addition of the rows with entries in the deleted column.

Figure 6 shows the binary computational tree using pivotal decomposition for the example problem of Eq. (1). The edge associated with variable $x_2$ is selected as pivot since its column count in the FDT is three, the smallest count that does not disconnect the graph. The all-terminal network state function is given by

\[ \Phi_{PC}(e) = e_i e_{i1} e_{i2} e_{i3} e_{i4} e_{i5} (e_{i6} \lor e_{i7}) + (1 - e_i) e_{i1} e_{i2} e_{i3} e_{i4} e_{i5} e_{i6} (e_{i7} \lor e_{i8}) \]

\[ \Phi_{PC}(e) = e_i e_{i6} e_{i7} e_{i8} e_{i9} (e_{i10} \lor e_{i11}) \]
(8)

where $(e_i \lor e_j) = 1 - (1 - e_i)(1 - e_j) = e_i + e_j - e_i e_j$. Note that $e_i(e_i \lor e_j) = e_i$.

The binary computational tree shown in Fig. 6 may also be used for computing the network resilience function. The network resilience function is given by

\[ \Phi_{PC}(e) = 6 e_i + 3 e_{i1} + 6 e_{i2} + e_{i3} e_{i4} e_{i5} + 2 e_{i6} e_{i7} + 3 e_{i8} + 3 e_{i9} + 6 e_{i10} + 6 e_{i11} + 3 e_{i12} + 3 e_{i13} + 2 e_{i14} e_{i15} + 2 e_{i16} e_{i17} + 2 e_{i18} e_{i19} + e_{i20} e_{i21} + e_{i22} e_{i23} + e_{i24} e_{i25} \]

(9)

Note that $\Phi_{PC}(1) = \binom{28}{2} = 28$, the number of connected pairs in the graph when every link is functioning.

**Optimal Partitioning Formulation**

The optimal decomposition problem is formulated as a multi-objective optimization problem with two conflicting objectives: (1) minimize the size of the master problem by minimizing the number of linking variables, and (2) minimize the size of subproblems by maximizing the number of partitions, these being partitions of similar size. To maximize the number of partitions, we minimize the network resilience. To minimize the number of linking variables, i.e., failed links, we maximize the sum of the indicator variables $e_i$. Hence, the mathematical formulation of the OMBP problem is as follows:

\[ \min_{e \in \{0, 1\}^m} \left\{ -\sum_{i=1}^m e_i \Phi_{PC}(e) \right\} \]

(10)

$m$ Pareto points $e^*$ may be obtained by minimizing $\Phi_{PC}$ subject to the successive constraints $\sum e_i = m, m - 1, \ldots, 0, 1$, over the space $\{0, 1\}^m$. Alternatively, as shown below under certain assumptions, a method of objective weighting or a greedy algorithm may be used to generate the Pareto solutions. Graphs with weighted edges may be considered by solving Eq. (10) after $e$ is replaced by $eW$, where $W$ is a $m \times m$ diagonal matrix of edge weights. In this case, edge weights depict strength of function-variable dependence or amount of transferred data between simulation modules. Groups of functions may also be constrained to a single partition by assigning large weights to corresponding incidence edges.

**Definition**

Let $S$ be a finite set and $f$ be a real-valued function defined on the subsets of $S$. If $f(X \cap Y) + f(X \cap Y) \leq f(X) + f(Y)$ for any two subsets $X$ and $Y$ of $S$, then $f$ is submodular on $S$. If the inequality is strict for all unordered $X$ and $Y$ in $S$, i.e.,
neither $X \subseteq Y$ nor $X \supseteq Y$, then $f$ is strictly submodular. If $f$ is submodular, then $-f$ is supermodular, $f$ is a modular if $f$ is both submodular and supermodular.

**Lemma 3 (from Table I, Topkis (1978))**

Let $g$ be a real-valued, supermodular, increasing or decreasing function on the subsets of $S$, and let $f$ be a real-valued, decreasing and concave function on the real line. Then $f \circ g$ is a real-valued submodular function on the subsets of $S$.

**Assumption**

There exists a real-valued, increasing function $\psi$ on the real line such that $\psi \circ \Phi_{PC}$ is submodular on $\{0, 1\}^n$.

**Basis:** Lovász (1983) in example 1.5 shows that for a graph $G$, the number of connected components of the subgraph $(V(G), X \subseteq E(G))$, $c(X)$, is a supermodular set function.

Function $c(\cdot)$ is decreasing. For the incidence vector $e$ of $X \subseteq E(G)$, let $c(e) = c(X)$. Since $\Phi_{PC}$ is a measure of graph connectivity, assume that $c = h \circ \Phi_{PC}$ for some real-valued, decreasing function $h$ on the real line. Let $f$ be a real-valued, decreasing and concave function on the real line. Then $f \circ (h \circ \Phi_{PC}) = (f \circ h) \circ \Phi_{PC}$ is submodular on $\{0, 1\}^n$.

We obtain the following result to define a greedy algorithm for the solution of the problem of Eq. (10) in which $e^*$ is increasing with respect to $\sum_{i=1}^{m} e_i^*$.

**Proposition 3**

Let $e^{*r}$ and $e^{*r'}$ be Pareto solutions to the problem of Eq. (10) such that $\sum_{i=1}^{m} e_i^{*r'} > \sum_{i=1}^{m} e_i^{*r}$. Let $\psi \circ \Phi_{PC}$ be strictly submo-

---

Because $c(X \cup \{e\}) - c(X), X \subseteq E(G) \setminus \{e\}$, are monotone increasing set functions in $X$ for all $e \in E(G)$.

---

**Proof:** $\psi \circ \Phi_{PC}$ is a submodular function on $\{0, 1\}^n$, for some increasing function $\psi$ on the real line. Thus, there exists a Lovász extension of $\psi \circ \Phi_{PC}$ to $\{0, 1\}^n$, namely $\langle \psi \circ \Phi_{PC} \rangle$, which is convex (Lovász, 1983). The cardinality of a set is a modular function. Therefore, every Pareto solution to the two-objective optimization problem of Eq. (10) may be obtained by solving the following substitute problem (regardless of $\psi \circ \Phi_{PC}$ being strictly submodular):

$$
\min_{e \in \{0,1\}^n} \{ \psi \circ \Phi_{PC}(e) - w \sum_{i=1}^{m} e_i \} \tag{11}
$$

where $w$ varies in $[0, \infty]$. The criteria space $\{\langle \psi \circ \Phi_{PC}(x), \sum_{i=1}^{m} x_i \rangle : x \in \{0, 1\}^n\}$ of the Lovász-extended problem is convex.

Also, the expression in brackets of Eq. (11) is submodular in $(e, w)$. From Topkis (1978), Theorem 6.1, the optimal solutions $e_i^*$ are increasing functions of $w^*$. Note (from Topkis, 1978, Theorem 4.1) that unless $\psi \circ \Phi_{PC}$ is strictly submodular, the solution set of Eq. (11), for a fixed $w$, may contain two or more optimal points $e^{*r}$ and $e^{*r'}$ such that neither $e^{*r} \neq e^{*r'}$ nor $e^{*r} \geq e^{*r'}$. From the convexity of the Lovász-extended problem, $\sum_{i=1}^{m} e_i^*$ is increasing with respect to $w$. So, $e_i^*$ is increasing with respect to $\sum_{i=1}^{m} e_i^*$ if $\psi \circ \Phi_{PC}$ is strictly submodular.

---

If $f$ is a function on $T$ and $T \subseteq S$, then a function $g$ on $S$ is an extension of $f$ to $S$ if $g(x) = f(x)$ for all $x$ in $T$.

If $\Phi_{PC}$ is a function on a set of products and $-w$ as a uniform unit price of each product, the submodularity of $-\Phi_{PC}$ implies that the products are complementary. That is, increasing the price of one product will not lead to an increase in the optimal levels of any product.
Solution of the Optimal Partitioning Problem

The problem of Eq. (10) may be solved by solving the substitute formulation of Eq. (11) with \( w \) as a parameter. Depending on the characteristics of the resilience function \( \Phi_{pc} \), we can distinguish the following cases:

(a) \( \Phi_{pc} \) is not submodular. Solution of Eq. (11), with \( \psi \) assumed the identity function, may generate only a subset of the Pareto set, whereas solution of Eq. (10) as specified in the paragraph following the equation generates the complete Pareto set. A global optimization technique is needed in both cases.

(b) \( \psi \circ \Phi_{pc} \) is submodular for some increasing function \( \psi \) on the real line. Solution of Eq. (11) generates the complete Pareto set. The problem is one of minimizing a submodular function on a collection of subsets of a finite set, a problem solvable in oracle-polynomial time (Grötschel et al., 1988). An integer programming technique is needed.

(c) \( \psi \circ \Phi_{pc} \) is strictly submodular for some increasing function \( \psi \) on the real line.\(^{10} \) The following greedy algorithm generates the complete Pareto set \( \{e^k\}_{k=0}^{\infty} \) for the OMBP problem. The number of linking variables increases after each iteration. At each step the variable that would result in the largest decrease in network resilience is selected as linking variable from the current set of local variables. Note that unless \( \psi \circ \Phi_{pc} \) is strictly submodular, the greedy algorithm may generate some feasible non-Pareto points and skip some Pareto solutions.

1. \textbf{initialize} \( e^0 = 1, k = 0 \)
2. \textbf{if} \( e^k = 0 \) \textbf{then stop}
3. \textbf{else find} \( i^* = \arg \max \{ \Phi_{pc}(1, e^k) - \Phi_{pc}(0, e^k) \text{ such that } e^k_i = 1 \} \)
4. \( (0, e^k) \rightarrow e^{k+1} \) (if more than one \( i^* \), branch on every different case)
5. \( k + 1 \rightarrow k \)
6. \textbf{return to (2)}

Table 1 shows the results from the above greedy algorithm for the problem of Eq. (1) and the resilience function given in Eq. (9). The first six rows of the table describe Pareto optimal partitions with 0, 1, ... , and 5 linking variables, respectively. A linking variable has zero as indicator variable \( e_i \), whereas for a local variable \( e_i \) is one. The partition shown in Fig. 2(b) is depicted by the second row in Table 1. A bold-faced number indicates information used to generate the next Pareto solution, according to step (3) of the greedy algorithm. For example, \( e_1 \) is first chosen as linking variable since the failure of its associated links causes the greatest decrease (=19) in the network resilience. Additional criteria, such as maximum number of linking variables or minimum number of disjoint partitions, may be further used to select a Pareto solution. Note that there is no Pareto optimal decomposition having \( x_5 \) as unique linking variable when the network resilience is used as connectivity measure; however, \((0, 1)\) is a Pareto point when the all-terminal network state is the connectivity measure. This is an example of how these two measures complement each other, network resilience being a more refined quantifier of network connectivity.

Conclusion

The formulation of the OMBP problem is based on a network reliability problem analogy. The relationships among design variables are the communication centers of a network. The design variables themselves are the communication links between these centers. The optimal decomposition problem is then reduced to one of finding the links that have the most effect on the overall network reliability and assigning their associated variables to a master problem, the top decision-maker. The rest of the design variables become local variables to be controlled by a number of low-level decision-makers at the subproblem level. The optimal decomposition is attained by minimizing the network resilience while maximizing the number of operating links in a Pareto sense.

The next step in this research is to develop additional metrics for problem partition and study different types of constraints that an OMBP problem formulation may require. For instance, existing design analysis or simulation capabilities may impose restrictions on the way a design problem is partitioned. Object partitioning or aspect partitioning also entails imposition of constraints on the OMBP problem formulation.

Acknowledgment

This work has been partially supported by a research grant from Ford Motor Company. This support is gratefully acknowledged.

References


