

TECHNICAL RESEARCH REPORT

A Multi-Objective Integer Programming Framework For Product Design

by Vinai S. Trichur, Michael O. Ball

T.R. 98-60



ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the Glenn L. Martin Institute of Technology/A. James Clark School of Engineering. It is a National Science Foundation Engineering Research Center.

Web site <http://www.isr.umd.edu>

A Multi-Objective Integer Programming Framework For Product Design

Vinai S. Trichur¹

Michael O. Ball²

Abstract

This paper describes a modeling framework for product design that facilitates the incorporation of both engineering and strategic considerations at the design stage. We first develop an abstract representation of a generic product, an AND/OR tree, that is context-independent and can be used to model a wide variety of products in different application settings. We show how this representation leads naturally to a mathematical model and discuss some of the properties of this model. Next, we show how the AND/OR tree can be employed in different settings; specifically, we describe applications to printed circuit assembly, and microwave module industries. These applications result in multiobjective integer programming formulations. We discuss the properties of these formulations, develop appropriate solution procedures, and report our computational experience. One of the advantages of the framework that we describe is the ease with which it can be extended to incorporate additional considerations. We indicate some possible extensions that might find ready applicability in industry.

1 Introduction

Over the past decade, two developments have had a profound impact on the way manufacturing firms operate. The first is the advent of the concurrent engineering paradigm, wherein various ‘downstream’ product life cycle considerations (such as cost, quality, manufacturability, etc.) are explicitly considered at the design stage; the prohibitive costs associated with the design-build-test-redesign loop no longer permit the designer to simply concentrate

¹Institute for Systems Research and College of Business and Management, University of Maryland, College Park, Maryland. Email: vtrichur@rhsmith.umd.edu

²Institute for Systems Research and College of Business and Management, University of Maryland at College Park. Email: mball@rhsmith.umd.edu

on product functionality at the design stage. The second is the impact global competition has had on firms - many industries have moved away from a highly vertically integrated to a more disintegrated or ‘leveraged’ model. In short, we are entering an era of ‘virtual manufacturing’, where complex products are designed and manufactured by widely dispersed groups of partners and suppliers.

These two developments have in turn spawned two distinct streams of research in the manufacturing literature. The first addresses the concurrent engineering problem by trying to develop models that facilitate ‘design for X’, where X might stand for cost, quality, manufacturability, and so forth. The second addresses the supply chain management problem via models that, given a firm’s product mix and manufacturing requirements, seek to determine optimal supplier and distribution configurations.

One assumption implicit in the above models is that the product design problem can be naturally *decomposed* into its two components; in other words, much of the research has been focused either on the concurrent engineering problem, or on the supply chain management problem, without taking into account the interactions between the two. We are of the opinion that this approach simply leads to another loop: the ‘design-strategic feasibility evaluation-redesign’ loop, wherein the designs generated by the concurrent engineering tools may not be compatible with the capabilities of the suppliers selected by the partner evaluation tools.

In a true virtual manufacturing environment the two approaches will need to be combined into a single, *integrated* model that lets designers take into account not only product life cycle considerations, but also supplier capability information, and information concerning the firm’s own strategic and financial goals.

There are two basic research issues underlying this problem—the first relates to the mathematical programming task while the second addresses the system integration task. We now briefly describe these issues in turn.

We need a mathematical modeling framework that, first, captures the *generic* structure of a product. In order to have the widest possible applicability, the model should be independent

of the specific application setting, i.e., it should simply make explicit the decisions involved in designing a product, without specific reference to the consequences of these decisions—these are modeled by metrics which might vary across industries, and, within an industry, across product lines. Next, the model should incorporate both engineering and strategic metrics (these would be tailored to reflect the objectives of the firm/application in question), and permit a study of the interactions between these metrics. Finally, the model should be flexible enough to permit applications to both *conceptual* design (a “macro-level” problem) and *detailed* design (a “micro-level” problem).

Once we have a mathematical model in place, we need to address the issue of data integration. Our model will require data from disparate sources. These sources may either be different databases within the same firm, or databases outside the firm, such as product catalogs from suppliers, data from financial clearing houses that track companies, etc. Clearly, integrating data from such vastly disparate sources and presenting them in a coherent form to the designer is a daunting systems integration task.

In this paper we describe a modeling framework for product design that satisfies the requirements enumerated above. We illustrate the framework via two applications—specifically, to the design of printed circuit board assemblies, and microwave modules. While the applications that we describe are more concerned with the “detailed-design” problem, the framework can be easily adapted to handle design problems of a more conceptual nature by using appropriate data to drive the model. We are currently building an integrated system in conjunction with a Fortune 500 firm, that implements the framework and addresses the data integration tasks.

2 Literature Review

Concurrent engineering and supply chain management are both very active areas of research. In this section we review some of the relevant literature, especially literature pertaining to the applications that we discuss in Sections 4 and 5.

There have been a number of efforts in the direction of concurrent engineering in industry, specifically via *integrated product and process design systems*. Initial efforts were focused on establishing guidelines to inform designers of manufacturing and assembly concerns to be addressed at the design stage (Bakerjian 1992, Boothroyd and Dewhurst 1983, Bralla 1986).

Using design for assembly guidelines, Jakiela and Papalambros (1989) built a rule-based Design-For-Assembly system that gives feedback about assemblability when the designer adds new features to the design. Gupta *et. al.*(1994) have developed IMACS, a system which generates the best operation plans for machined components and gives feedback about manufacturing infeasibilities in the design. However, none of these tools are applicable to the electronic domain.

In the electronic domain, Harhalakis *et. al.*(1993) have developed a rule-based system for critiquing the manufacturability of microwave modules. Feldmann and Frank (1993) describe a system that integrates electronic and mechanical CAD tools. These tools do not evaluate the designs with respect to cost and lead times.

Schemes for costing a given microwave module design can be found in a number of sources (see, for instance, Heng and Gay 1991, Oh and Park 1993). Reviewing this body of literature reveals that attempts to determine optimal designs (rather than assessing a given design) with respect to cost have been rather limited. Oh and Park (1993) use a dynamic programming approach to optimize the assembly processes; however, their procedure does not appear to be very practical for situations having a large search space of design alternatives. The only other optimization application we have come across is Russell (1986).

An excellent review of the supply chain management literature can be found in Thomas and Griffin (1996). The literature may be broadly classified into two categories. The first addresses operational planning issues, such as the determination of optimal reordering policies. The second category is of more interest to us, and addresses strategic planning issues, such as vendor and plant location selection. A comprehensive integer programming based model can be found in Arntzen, *et. al.*(1995). Other applications include Cohen and Lee

(1989), and Brown, *et. al.*(1987).

All of the above models assume that the product mix is known, i.e., they assume that the product design task has been completed. We are not aware of any models that jointly consider product design and supplier selection issues.

3 Modeling a Hierarchical System

We begin by introducing an abstract model that captures the *structure* of *any* hierarchical system, and makes explicit the *decisions* involved in designing such a system. This abstract model, the *AND/OR tree*, will constitute the core of all our subsequent models of manufacturing systems. Next, we show how an AND/OR tree may be modeled as a system of equations. We conclude with an investigation of an important property of this system of equations.

A *hierarchical system* is one which can be naturally decomposed in a top down fashion into subsystems, subsystems, and so forth. The decomposition process continues until we encounter *atomic elements* that cannot be broken down further. Figure 1(A) illustrates this concept. This representation of a hierarchical system captures the essence of most of the complex systems that are routinely encountered. We will refer to this representation as an *AND tree* (reflecting the fact that the system *A* contains subsystems *B* and *C*, and so forth.)

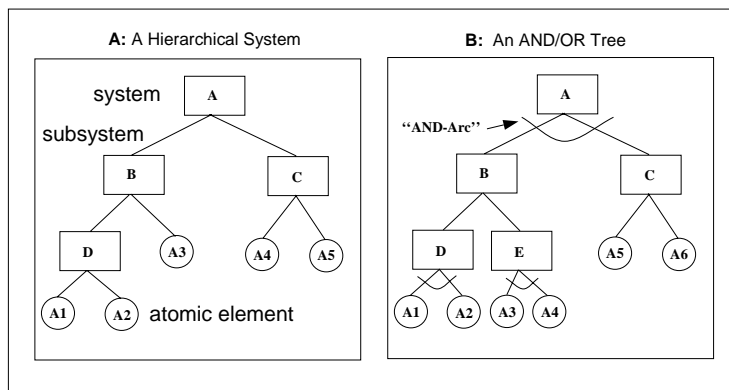


Figure 1: Modeling a hierarchical system via an AND/OR Tree.

A natural way to incorporate *alternatives* (for some/all of the subsystems/atomic elements) into the above representation is to model it as an *AND/OR tree*. Figure 1(B) illustrates this concept. Here, the system, A , contains subsystems B and C (indicated by the “AND-arc”). B can be decomposed further in two *alternative* ways; thus, B contains *either* subsystem D , *or* subsystem E . D contains atomic units A_1 and A_2 , E contains A_3 and A_4 , and C contains either A_5 or A_6 . AND/OR trees are a special case of more general structures, AND/OR graphs. These structures find applications in problems such as theorem proving, symbolic integration, and analysis of industrial schedules (see Horowitz, Sahni, and Rajasekaran 1998). We will comment more on the relationship between AND/OR trees and AND/OR graphs in Section 4.3.

Since designing a system involves making *choices* among the available alternatives, we will now show how the AND/OR tree can be modeled as a system of equations that can be embedded into mathematical programming based optimization algorithms. First, we note that each of the nodes constituting the AND/OR tree is either an AND-node, whose selection necessitates the selection of *all* of its child nodes, or an OR-node, whose selection necessitates the selection of *exactly one* of its child nodes. In Figure 1(B), A , D and E are AND-nodes, while B and C are OR-nodes. It is possible that in some applications the selection of a node will call for the selection of a *subset* of its child nodes (the ‘AND-children’) and exactly one of the remaining child nodes (the ‘OR-children’). We observe that such an AND/OR tree can always be transformed into one containing only AND-nodes and OR-nodes (the standard form), by the introduction of appropriate *dummy* nodes. The transformation essentially mimics the process whereby an arbitrary boolean expression is transformed into the conjunctive normal form. We will assume henceforth that the AND/OR tree is of the standard form. Letting \mathcal{V}_{AND} = the set of AND-nodes, \mathcal{V}_{OR} = the set of OR-nodes, $\mathcal{V} = \mathcal{V}_{AND} \cup \mathcal{V}_{OR}$ = the set of nodes in the tree, $ROOT$ = the root node of the tree, and, $CHILD(i)$ = the set of nodes that are children of node i , we can define the following decision variables:

$$x_i = \begin{cases} 1 & \text{if node } i \text{ is selected to be in the system, } i \in \mathcal{V}, \\ 0 & \text{otherwise.} \end{cases}$$

The constraints describing the AND/OR tree would be as follows:

$$\begin{aligned}
 x_{ROOT} &= 1 \\
 x_j &= x_i \quad \forall i \in \mathcal{V}_{AND}, \quad j \in CHILD(i) \\
 \sum_{j \in CHILD(i)} x_j &= x_i \quad \forall i \in \mathcal{V}_{OR}
 \end{aligned}$$

The first constraint ensures that the root node is always selected, the second set of constraints (the AND-constraints) ensures that if an AND-node is selected, all of its children are also selected, and, finally, the third set of constraints (the OR-constraints) ensures that if an OR-node is selected, exactly one of its children is also selected. Given a design problem, the above equations will appear as constraints in the corresponding mathematical programming formulation of the problem. Since the x_i variables are binary, we would be dealing with an integer program. We will now show that the above system of equations possesses a useful property, total unimodularity, that will be exploited in subsequent sections. Total unimodularity of the constraint matrix associated with an integer program permits us to solve the problem using linear programming algorithms (if the right hand side vector associated with the constraint matrix is integral, as it is in the above case, and if the objective functions are linear, as they are in the applications that we describe), thus leading to extremely efficient solution methods.

Definition 1 *An $m \times n$ integral matrix A is totally unimodular (TU) if the determinant of each square submatrix of A is equal to 0, 1, or -1.*

It is evident that $a_{ij} = 0, 1, \text{ or } -1$ if A is TU; that is, A is a $(0, 1, -1)$ matrix. Using certain properties of TU matrices, it is possible to prove the following theorem (the proof appears in the appendix):

Theorem 1 *The constraint matrix associated with any AND/OR tree is Totally Unimodular.*

4 Application 1: Design of Printed Circuit Board Assemblies

The first application that we describe involves the design of printed circuit board assemblies, specifically transmitter/receiver (T/R) modules. This work is described in Ball *et. al.*(1995). In this application we show how the AND/OR tree framework that was developed in the preceding section can be employed to model a T/R module. The specific metrics that we seek to optimize are a cost metric and a manufacturing yield metric (defined as the product of process yields and component defect rates). It should be noted that the yield associated with a given design could be factored into the analysis via a rework cost. However, we view cost and yield as competing (and often conflicting) evaluation metrics for which a *multiobjective* approach seems to be more appropriate. In particular, yield can be viewed as a measure of manufacturing quality, which leads to its consideration as a metric independent of cost.

4.1 Modeling a T/R Module

Any electromechanical or electronic product is designed to satisfy a certain *function*; for instance, a T/R module (which is a basic component of most radar systems) should transmit and receive radio messages. This basic function can then be decomposed into subfunctions, which can then be recursively decomposed further. These *function blocks* are, thus, abstract representations of what a product *must do* in order to accomplish its function—it is at this level that design innovation usually takes place. Given a function block representation of a product, it would then be possible for designers to postulate alternate function blocks that accomplish the same function.

The decomposition process continues until the function blocks become ‘concrete’ enough, i.e., until it becomes possible to map a function block on to an assembly/component that can be manufactured/purchased. Figure 2(A) illustrates this idea. Each of the terminal assembly nodes in Figure 2(A) can be decomposed into their constituent components. Each of these components will have alternatives; moreover, each component will have a set of processes that need to be performed on it, and each of these processes will also have alternatives. Figure 2(B)

shows this decomposition of an assembly into its constituent components and processes. Once we develop such a decomposition for each terminal assembly node in Figure 2(A), we will have a complete AND/OR tree description of the product. Our objective is to choose among the alternative function blocks, assembly blocks, components, and processes for each component, such that the resulting design is ‘optimal’ with respect to the cost and manufacturing yield metrics.

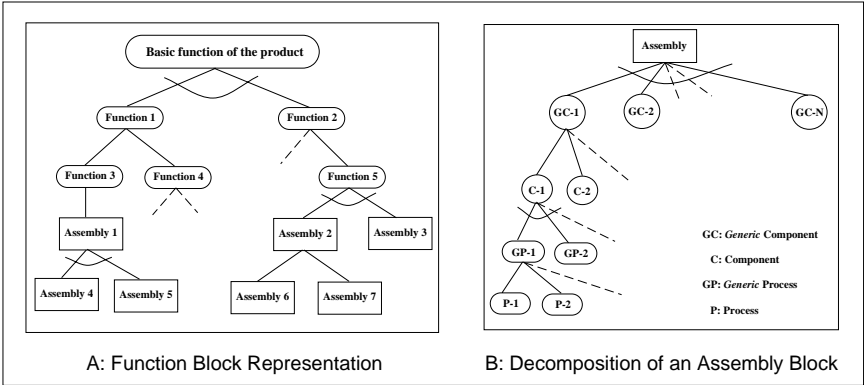


Figure 2: Decomposition of a Product.

We now state the three key assumptions underlying our model. First, we assume that sub-assemblies are manufactured independently. This assumption would not be valid if, for instance, different sub-assemblies could be acted upon simultaneously during a single setup of a process. Second, we do not consider the impact of commonality of components across sub-assemblies. In some manufacturing situations, it might be advantageous to use common (i.e., identical) components in different sub-assemblies, even though the sub-assemblies may be manufactured separately (a simple example would be quantity discounts). Third, the two metrics that we consider, cost and yield, are *decomposable* metrics, in the sense that the cost/yield contribution of an assembly block is assumed to depend *upon that block only*. Of course, this would no longer be true without assumptions (1) and (2). These three assumptions permit us to decompose the product into its constituent assemblies—we determine the optimal choice of components and processes for each subassembly and then construct the overall solution using these partial solutions. We examine the consequences of relaxing these

assumptions in subsequent sections.

4.2 The Integer Programming Formulation

We now describe the optimization problem for a single assembly. By our assumption of independence of the assemblies constituting the product, the final solution for the entire product is simply a concatenation of the solutions for the individual assemblies. We assume that the input data for the problem consists of the quantities defined in Table 1.

Table 1: Notation

\mathcal{V}	=	set of generic components,
\mathcal{V}_j	=	set of alternatives for the j th generic component,
\mathcal{P}	=	set of all processes (including alternate processes),
\mathcal{P}_j	=	set of ‘generic’ processes related to the j th component,
\mathcal{P}_{ji}	=	set of alternatives for the i th generic process related to the j th component: $i \in \mathcal{P}_j$,
c_j	=	unit cost of j th component: $j \in \mathcal{V}_k, k \in \mathcal{V}$,
t_p	=	setup time of p th process: $p \in \mathcal{P}$,
t_{pj}	=	runtime when p th process is used for j th component: $j \in \mathcal{V}_k, k \in \mathcal{V}, p \in \mathcal{P}_{ji}, i \in \mathcal{P}_j$,
α_j	=	defect rate of j th component: $j \in \mathcal{V}_k, k \in \mathcal{V}$,
β_p	=	yield rate of p th process: $p \in \mathcal{P}$,
l	=	labor cost per unit time,
b	=	batch size.

We now define the following decision variables:

$$x_j = \begin{cases} 1 & \text{if component } j \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

$$y_p = \begin{cases} 1 & \text{if process } p \text{ is used in the assembly,} \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{pj} = \begin{cases} 1 & \text{if process } p \text{ is selected for component } j, \\ 0 & \text{otherwise.} \end{cases}$$

The expressions for cost and yield are given as follows:

$$C = \text{Unit cost} + \text{Runtime cost} + \text{Setup cost} = \sum_i c_i x_i + l \sum_{p,j} t_{pj} x_{pj} + \frac{l}{b} \sum_p t_p y_p \quad (1)$$

$$Y = \prod_p (\beta_p)^{y_p} \prod_j (1 - \alpha_j)^{x_j} \quad (2)$$

We note that the labor cost, l , might be different for manual and automatic processes; thus, the runtime cost may consist of two terms, one for manual processes and another for automatic processes. The yield expression, (2), consists of the product of the component defect rates and process yields. We can now linearize (2) to get

$$Y' = \log Y = \sum_p y_p \log \beta_p + \sum_j x_j \log(1 - \alpha_j). \quad (3)$$

The problem we wish to solve is the following multi-objective integer program:

$$\text{minimize} \quad \left\{ \begin{array}{c} C \\ -Y' \end{array} \right\}$$

subject to

$$\sum_{j \in \mathcal{V}_k} x_j = 1 \quad k \in \mathcal{V} \quad (4)$$

$$\sum_{p \in \mathcal{P}_j} x_{pj} = x_j \quad \forall j, \quad i \in \mathcal{P}_j \quad (5)$$

$$y_p \geq x_{pj} \quad \forall p, j \quad (6)$$

$$x_j, y_p, x_{pj} \in \{0, 1\} \quad \forall j, p \quad (7)$$

Constraints (4) and (5) capture the AND/OR tree structure of the problem. Constraints (6) tell us which processes have been selected.

4.3 Discussion

It is well known that solutions to the *parametric* problem **P**:

$$\begin{aligned} &\text{minimize} && Z(\lambda) = \lambda C - (1 - \lambda)Y', \\ &\text{subject to} && \text{constraints (4)-(7),} \end{aligned} \quad (8)$$

where the parameter λ ranges over the interval $[0, 1]$, are also *efficient* (Pareto optimal) solutions for the bicriteria IP problem (see, for instance, Gass 1985). It is this version of the problem that we shall address in what follows. It is straightforward to establish the following theorem by showing that the *uncapacitated facility location problem* is a special case of **P** (for a proof, see Ball, *et. al.* 1995).

Theorem 2 *Problem \mathbf{P} is NP-Hard.*

The facility location problem has been relatively well studied and a number of solution strategies have been reported in the operations research literature (see, for example, Nemhauser and Wolsey 1988). An important issue in our application here is that we would like to generate a *set* of Pareto optimal solutions parameterized with respect to λ . Hence, in selecting a solution procedure, we need to consider the ease with which parametric analysis can be carried out.

The solution procedure that we propose arises from the observation that the number of process alternatives involved in PCB assembly design at the manufacturing facility motivating this application is quite small. The small number of possible processes implies that an approach which starts by enumerating all possible process combinations (y vectors) is computationally feasible. We then note that for a given set of *selected processes*, that is, a set \mathcal{P}' such that

$$y_p = \begin{cases} 1 & \text{if } p \in \mathcal{P}', \\ 0 & \text{otherwise,} \end{cases} \quad (9)$$

problem \mathbf{P} becomes easy to solve. While decision problems on general AND/OR graphs are known to be NP-complete (see Horowitz, Sahni, and Rajasekaran 1998), it is easy to show that a polynomial-time greedy algorithm produces an optimal solution (see Ball, *et. al.* 1995) for the above problem. The greedy approach however does not permit a straightforward procedure for parametric analysis with respect to λ . For this purpose, consider the following reduced problem $\mathbf{P}(\mathcal{P}')$

$$\begin{aligned} & \text{minimize} && \lambda C - (1 - \lambda)Y', \\ & \text{subject to} && \text{constraints (4)-(7), and (9).} \end{aligned}$$

Now, constraints (4) and (5) are simply the AND/OR tree constraints, and, by Theorem 1, define a totally unimodular matrix. It is a well known fact (for instance, see Proposition 2.2 on page 541 of Nemhauser and Wolsey 1988) that if the constraint matrix for an integer programming problem, P , is TU, then the feasible region for this problem is an integral

polyhedron, i.e., has integer extreme points, as long as the right hand side is integer. Consequently, the linear programming relaxation of $\mathbf{P}(\mathcal{P}')$ is guaranteed to produce an integer solution. Hence $\mathbf{P}(\mathcal{P}')$ can be solved using standard LP software and full parametric analysis with respect to λ can also be efficiently performed. Furthermore, since all the variables are binary, the extreme points of the above polyhedron are the only integer feasible points for the problem $\mathbf{P}(\mathcal{P}')$. This fact enables us to use a well known result from linear vector maximization theory (see Theorem 5 on page 128 of Gass 1985, Philip 1972) to guarantee that parametric analysis with respect to λ generates *all* the efficient solutions for the problem $\mathbf{P}(\mathcal{P}')$.

In summary, our approach is to solve $2^{|\mathcal{P}'|}$ subproblems, one for each choice for \mathcal{P}' . For each subproblem, the optimal objective function value is obtained as a piece-wise linear function of λ , using linear programming sensitivity analysis techniques. Figure 3(A) shows such a parametric curve for one particular choice of \mathcal{P}' , say, \mathcal{P}'_1 . The problem $\mathbf{P}(\mathcal{P}'_1)$ has three efficient solutions, the first of which is valid (i.e., minimizes $Z(\lambda)$) in the range $0 \leq \lambda < \lambda_1$; similarly, the second and third solutions are valid in the ranges $\lambda_1 \leq \lambda < \lambda_2$, and $\lambda_2 \leq \lambda \leq 1$ respectively. The lower envelope of the family of these functions yields the parametric solution to the original problem \mathbf{P} . For instance, in Figure 3(B), the problem \mathbf{P} has three efficient solutions. When $0 \leq \lambda < \lambda_1$, it is optimal to select $\mathcal{P}' = \mathcal{P}'_2$, and the corresponding efficient solution obtained from the parametric curve for problem $\mathbf{P}(\mathcal{P}'_2)$ will be the one that minimizes $Z(\lambda)$ for problem \mathbf{P} . Similarly the appropriate efficient solutions for problems $\mathbf{P}(\mathcal{P}'_4)$ and $\mathbf{P}(\mathcal{P}'_3)$ are valid for problem \mathbf{P} when $\lambda_1 \leq \lambda < \lambda_2$, and $\lambda_2 \leq \lambda \leq 1$ respectively. Such a lower envelope can be constructed efficiently using standard computational geometry algorithms (see, for instance, Preparata and Shamos 1985).

We are now in a position to aggregate the results for the assemblies in order to obtain the solution for the entire product. For assemblies that are alternatives for each other, we simply take the lower envelope of their parametric curves (similar to Figure 3(B)); once this has been done, we add all the parametric curves. Figure 4(A) illustrates this process for a product

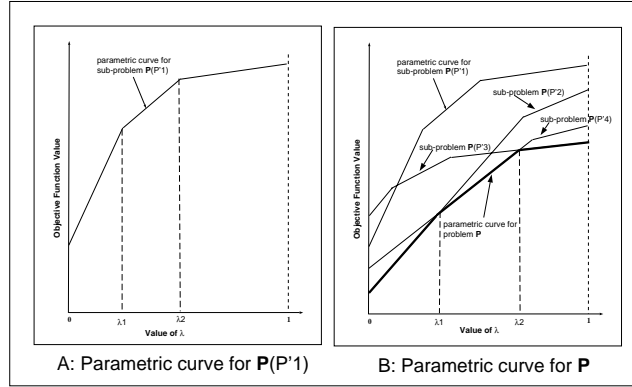


Figure 3: Parametric Analysis With Respect To λ .

that has two subassemblies. Once we have the parametric curve for the entire product, we can recover the cost and yield figures corresponding to each Pareto optimal solution, and plot the *tradeoff curve* for the product. This figure provides information relating to the marginal rate of substitution between cost and yield, and can be used by managers as an aid in decision-making.

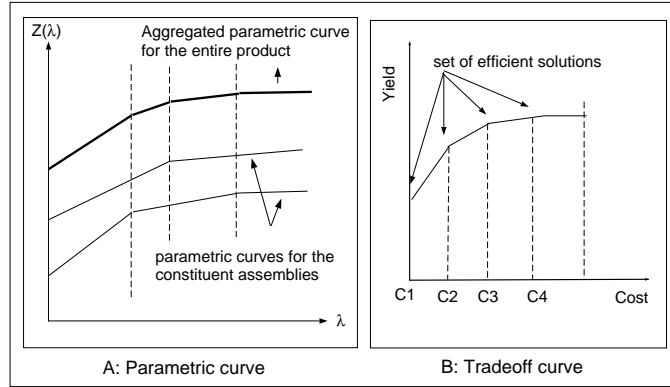


Figure 4: Parametric Curve For The Entire Product.

5 Application 2: Design of Microwave Modules

In this section we describe the application of our basic modeling framework to the design of microwave modules (see Trichur, *et. al.* 1996), electronic devices that are used in many modern telecommunications systems. This application is quite similar to the previous one—

we are interested in selecting components, and processes for each component, from a set of alternate processes and components. Unlike in the earlier application, microwave modules consist of only one assembly/board. Thus, the problem is simpler in some sense, since we no longer have to aggregate the solutions for individual assemblies. However, it is considerably more difficult than the single assembly problem (problem **P**) that we discussed in Section 4.3, due to the following reasons. First, the number of processes is much larger, and, consequently, the enumeration based approach discussed earlier is no longer viable. Second, we relax the assumption of independence of part choices, i.e., there is now an advantage to commonality of parts. Third, we now introduce strategic considerations into our model via additional metrics. Hence, the parameterization that we carried out in order to obtain **P** is no longer valid. Finally, we use more accurate estimates for process runtime costs and yields. This involves interaction with a process planner.

5.1 Problem Definition

Key attributes such as material costs, run times, setup times, process yields, and material defect rates are assumed to be known for components, processes, and component-process combinations. In addition, we assume that for each component, we know the supplier associated with that component, and the delivery lead time of that supplier. The problem is to determine a set of components (and implicitly, suppliers) and processes that are ‘efficient’ with respect to four objectives—cost, yield, supplier lead time, and number of suppliers used.

Let \mathcal{S} = the set of suppliers, \mathcal{S}_j = set of components supplied by j th supplier, and, d_s = delivery lead time of s th supplier. We now have the supplier variables,

$$w_j = \begin{cases} 1 & \text{if supplier } j \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

The expressions for cost and yield are exactly the same as before (equations (1) - (3)). We now discuss the other metrics—these incorporate some strategic considerations into the model and lead to an integrated concurrent engineering-strategic feasibility evaluation system.

The first new objective that we consider is supplier lead time. We would like to choose components (and hence suppliers) such that the delivery lead time is as low as possible. The overall delivery lead time, defined as the maximum of the lead times of the selected suppliers, is given by:

$$\text{Lead Time} = L = \text{Max}(d_1w_1, d_2w_2, \dots, d_Sw_S) \quad (10)$$

We linearize (10) by replacing L with auxiliary continuous variable, L' , in the objective function, and adding the following constraints:

$$L' \geq d_iw_i \quad \forall i \in \mathcal{S} \quad (11)$$

We would also like to minimize the total number of suppliers. Typically there are quantity discount advantages associated with ordering more components from the same supplier; also using a smaller number of suppliers reduces overhead costs related to inventory management and component tracking. There is also a substantial body of literature that puts forth theoretical arguments in support of a smaller network of suppliers, and presents empirical evidence for the prevalence of such networks in industry (see, for instance, Dyer 1990). Minimizing the total number of suppliers captures these effects. The number of suppliers is given by:

$$N = \sum_{i=1}^S w_i \quad (12)$$

The problem we wish to solve is the following multi-objective integer program:

$$\begin{aligned} & \text{minimize} \quad \left\{ \begin{array}{c} C \\ -Y' \\ L' \\ N \end{array} \right\} \\ & \text{subject to} \end{aligned}$$

$$\sum_{j \in \mathcal{V}_k} x_j = 1 \quad k \in \mathcal{V} \quad (13)$$

$$\sum_{p \in \mathcal{P}_{ji}} x_{pj} = x_j \quad \forall j, i \in \mathcal{P}_j \quad (14)$$

$$y_p \geq x_{pj} \quad \forall p, j \quad (15)$$

$$w_i \geq x_j \quad \forall i \in \mathcal{S}, j \in \mathcal{S}_i \quad (16)$$

$$x_j, y_p, x_{pj}, w_s \in \{0, 1\} \quad \forall j, p, s \quad (17)$$

Constraints (13) and (14) capture the AND/OR tree structure of the problem, and are identical to (4) and (5). Constraints (15) and (16) tell us which processes and suppliers have been selected. We would also need to include the set of constraints (11), that define the lead time variable, L' .

5.2 Discussion

Unlike in Section 4.3, we cannot eliminate the ‘facility location’ constraints, (15) and (16). Hence, we can no longer solve the LP relaxation and be guaranteed of integer solutions. However, this formulation of the uncapacitated facility location problem is known to be “strong”, meaning that IP solvers generally perform well on it, at least for problems of moderate size. As before, we are interested in finding *non-dominated* (‘efficient’, Pareto optimal) solutions. Also, since the set of efficient solutions may be very large, we feel that it would make more sense for the optimization to proceed in an interactive manner, with the designer controlling the ‘search direction.’ While there exists a large body of literature on multiobjective optimization (see, for instance, Serafini 1985, Goicoechea, *et. al.* 1992), most of this work is applicable only to convex search spaces, and when nonconvexities exist, the proposed algorithms are problem-specific. Consequently, we propose two alternative solution procedures that are generic enough to be used irrespective of the extensions that might be incorporated into the model.

The overall approach we use was motivated by the architecture used by the CONSOL system, which performs nonlinear, multiobjective optimization (Tits and Ma, 1986). Any mechanism for generating a *set* of efficient solutions requires some way of assigning *relative weights* to the objectives (the parameter λ served this purpose in Section 4.3). Consequently, we first require the user to specify a set of *good* and *bad* values for each of the objectives,

and then *normalize* the objectives, as follows:

$$Z_{i,norm} = \frac{Z_i - Z_{i,g}}{Z_{i,b} - Z_{i,g}} \quad (18)$$

where Z_i stands for C, Y', L' , or N , and $Z_{i,g}, Z_{i,b}$, and $Z_{i,norm}$ are the *good*, *bad*, and *normalized* values of the i -th objective. The *good* and *bad* values can be thought of as defining a range within which the objective is required to lie—for instance, we might want the yield to lie between 95% and 100%. The objective is not permitted to exceed its *bad* value (assuming that we are minimizing the objective.) Clearly, the narrower the range, the more difficult it becomes to restrict the objective within the range—objectives with narrow ranges are consequently given more weight during the optimization process. Thus, the user can alter the relative weights of the objectives by changing their *good* and *bad* values. Such an approach has a particularly intuitive appeal to users and was used as the basis for a graphical user interface designed to address these problems (Splain 1998). We now describe our two solution approaches.

5.2.1 Approach 1

In what follows, let $Z = (Z_1, \dots, Z_N)$ be the set of criteria/metrics under consideration, and assume without loss of generality that all the criteria have to be minimized. We define a new variable, $Z = \text{Max}(Z_{1,norm}, \dots, Z_{N,norm})$, and minimize Z . Thus, we would be minimizing the *maximum deviation* of an objective from its *good* value. Minimax optimization is a natural setting for obtaining nondominated solutions and is intuitively appealing, given the definition of Pareto optimality. Moreover, it is possible to show (by a straightforward application of Theorem 13.2 on page 324 of Brayton and Spence 1980) that this approach is guaranteed to generate *all* the efficient solutions to the IP problem.

One drawback to this approach, however, is that in addition to nondominated solutions, it is also capable of generating dominated solutions. The following simple example illustrates this fact. Let the criterion vector be $Z = (C, N, L')$, where the C, N and L' have their usual meaning. Let the solution obtained by solving the IP be $Z^* = (100, 5, 40)$, and let the good

and bad values that resulted in this solution be such that $Z_{norm}^* = (0.3, 0.5, 0.4)$, i.e., $\text{Max} \{Z_{i,norm}^*\} = 0.5$. Now, consider another feasible solution, $Z' = (100, 5, 30)$; this solution has the same cost and uses the same number of suppliers as the earlier solution, but has a lower maximum lead time, corresponding to the fact that a different set of five suppliers was chosen. Z' dominates Z^* . Thus, it is clear the optimal solution to the integer program might actually be a dominated solution. However, note that $Z'_{norm} = (0.3, 0.5, k)$, where $k < 0.4$ as $Z'_{L'} < Z^*_{L'}$. Consequently, $\text{Max} \{Z'_{i,norm}\} = 0.5 = \text{Max} \{Z_{i,norm}^*\}$. We formalize the above example in the following proposition.

Proposition 1 *Let Z^* be the solution that minimizes $\text{Max} \{Z_{i,norm}\}$. Suppose \exists a feasible solution Z' that dominates Z^* . Then, $\text{Max} \{Z'_{i,norm}\} = \text{Max} \{Z_{i,norm}^*\}$.*

Proof: By the optimality of $\text{Max} \{Z_{i,norm}^*\}$, we have $\text{Max} \{Z'_{i,norm}\} \geq \text{Max} \{Z_{i,norm}^*\}$. Since Z' dominates Z^* , we have $\text{Max} \{Z'_{i,norm}\} \leq \text{Max} \{Z_{i,norm}^*\}$. \square

Thus, if the solution returned by the IP (corresponding to the chosen *good* and *bad* values) is not a nondominated solution, the solutions that dominate it will appear as alternate optimal solutions to the same IP. Ideally, we would like to examine all the alternate optimal solutions to the IP and select only the nondominated ones; however, determining alternate optima for integer programs is not an easy task, as it is in the case of linear programs.

To summarize, although minimizing the maximum deviation of an objective from its *good* value is an intuitively attractive approach that is capable of generating all the efficient solutions, it does not guarantee nondominated solutions. Consequently, we propose the following alternative solution procedure.

5.2.2 Approach 2

We first define the auxiliary variables, $z_{i,n} = \text{Max}(Z_{i,norm}, 0)$. Now we let $z = \sum_{i=1}^N z_{i,n}$, and minimize z subject to $Z \leq 1$. The intuition behind this approach is that we might not mind one of the objectives being close to its *bad* value, if another objective were to be

correspondingly close to its *good* value—this can be modeled by minimizing the sum of the normalized objectives. However, we would have to ensure that one of the objectives does not become better than its *good* value at the expense of another objective—the definition of $z_{i,n}$ ensures this; essentially, there is no incentive to improve an objective beyond its *good* value (this would cause $Z_{i,norm}$ to become negative). The constraint $Z \leq 1$ ensures that no objective becomes worse than its *bad* value. It should be noted that for some choices of the *good* and *bad* values, this constraint might cause the IP to become infeasible, and, consequently, might need to be relaxed.

In order to characterize the solutions generated by this procedure, we first assume that *good* value for each objective is set to *optimal* value of that objective, i.e., we assume that $Z_{i,g} = Z_{i,opt}$, where $Z_{i,opt}$ is the value of objective i when *it alone* is minimized. This ensures that $Z_{i,norm}$ can never take on a negative value, and, since the equations defining $z_{i,n}$ are now not needed, we will have $z = \sum_{i=1}^N Z_{i,norm}$. Since z is now simply a weighted sum of the objectives, it is easy to show (see Theorem 5 on page 227 of Gass 1985) that minimizing z is *guaranteed* yield a nondominated solution. However, we note that this result will not hold for arbitrary *good* values (i.e., *good* values not equal to the single objective optima); in this situation, it is possible that the solution obtained by solving the IP may be dominated by alternate optimal solutions to the same IP (a result similar to Proposition 1). Furthermore, this approach will not yield the entire set of efficient solutions (see Theorem 11.2 on page 275 of Brayton and Spence 1980); it will do so only if the optimization is carried out over the convex hull of integer points.

In summary, we have outlined two solution procedures for solving the multiobjective integer programming formulation of the product design problem. In either case, the optimization will proceed in an interactive fashion, with the user modifying the *good* and *bad* values at each stage, to yield a set of solutions.

5.3 Computational Experience

In order to test the strength of the formulation and verify that the solution procedures outlined above are viable, we generated a set of test problems and conducted a series of experiments using these problems. A typical microwave module board contains between 25 and 100 components. Consequently, we generated four sets of test problems. The first set consisted of boards containing 25 components, while the second, third, and fourth sets consisted of boards containing 50, 75, and 100 components on the board respectively. Within each set, we had five test problems, yielding a total of twenty test problems. The components on the boards had between three and six alternatives each. We modeled ten processes, which had between two and five alternatives each. Both the component and process data were based on figures supplied by the manufacturing facility that motivated this work. The components were supplied by ten vendors, each having a different delivery lead time. Table 2 summarizes the relevant statistics for the the integer programs corresponding to the above set of problems. For each set, we report the mean number of variables, constraints, and constraint nonzeroes, in the integer programs corresponding to the problems in that set. It is clear that as the number of parts on the board grows, we are faced with integer programs of increasingly non-trivial size.

Table 2: Test problem statistics

Problem Size	Variables	Constraints	Nonzeroes
25 Parts	1179	1521	4362
50 Parts	2518	3600	9405
75 Parts	3836	5493	14350
100 Parts	5052	7268	18946

Our first set of experiments consisted of optimizing each objective individually for each of the above problems. The optimal values of the individual objectives are required for testing the second approach that we outlined above for solving the multiobjective problem. All of our experiments were carried out on a Sun SPARC 10 workstation running CPLEX version 4.0. All the solution times were of the order of a few seconds, and, consequently, deemed

acceptable for an interactive solution procedure.

Next, we examined the effect on solution time, of increasing the number of objectives in the multiobjective model—we would expect the solution time to increase as the number of objectives goes up. Figure 5 summarizes the results of this experiment. The first observation of note is that the second solution procedure that we outlined above significantly outperforms the first. Next, we note that the solution time does not appear to be sensitive to the number of objectives in the model, for smaller problem sizes. Finally, we observe that while the solution time does increase with the number of objectives in most cases, there are some situations where it actually decreases as the number of objectives rises.

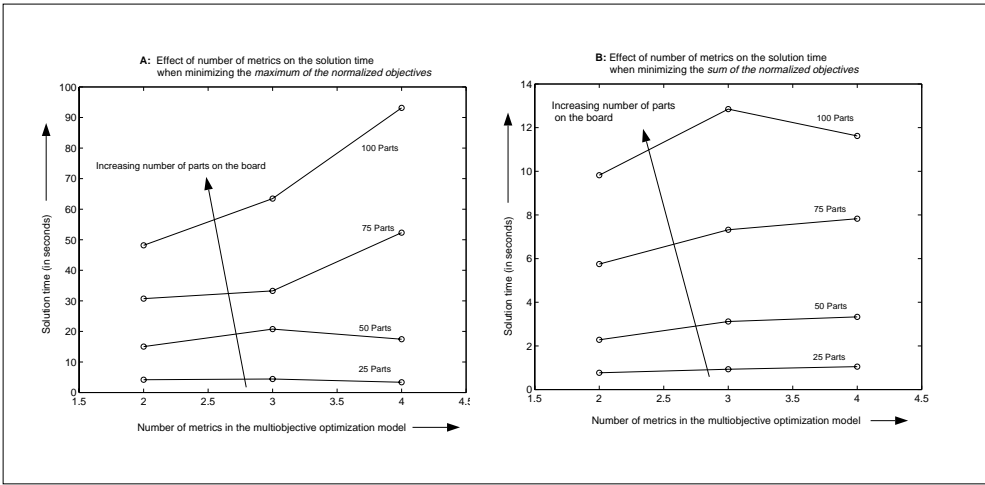


Figure 5: Solution time vs. number of metrics in the multiobjective model.

Our final series of experiments was designed to determine how sensitive the solution times were to variations in the *good* and *bad* values. Intuition would seem to suggest that the ‘tighter’ the allowable ranges (defined by the *good* and *bad* values) for the metrics, the larger would be the solution time, since the constraints on the metrics are now ‘more difficult to satisfy’ in some sense. There is, however, a flip side to this argument—if the allowable ranges on *all* the metrics are very tight, then the number of feasible solutions satisfying these ranges would be quite small. Thus, we would expect to see the following behavior as we successively tighten the allowable ranges for an increasing number of metrics—the solution time should

first increase for a while, and then decrease. Figure 6 summarizes the results of this series of experiments, for the case where we have three objectives in the model. The figure on the left plots the solution time vs. ‘tightness of the allowable ranges’ for the first solution approach, while the figure on the right plots the results for the second. Again, we note that the second approach considerably outperforms the first. Next, we observe that the solution times do seem to exhibit the behavior that we intuitively expected. Also, the solution times appear to be relatively insensitive to the tightness of the allowable ranges, for lower problem sizes. Finally, the second approach appears to be more ‘stable’, in the sense that the solution times do not exhibit too great a variation—a characteristic that would definitely be desirable in an interactive system.

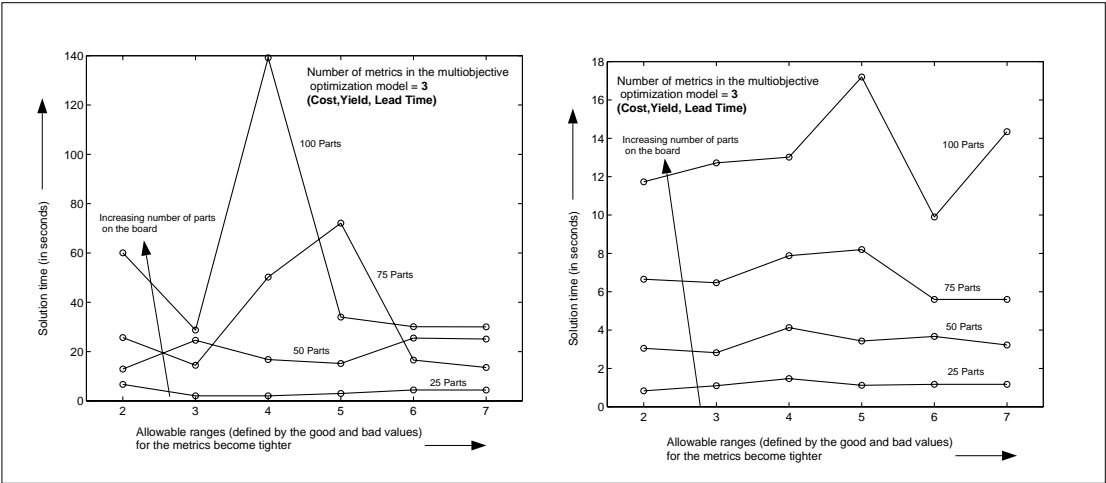


Figure 6: Solution time vs. tightness of allowable ranges for the metrics.

To summarize, our experiments appear to indicate that the solution procedures outlined above are viable, even for large problem sizes, and, therefore, can be incorporated into an interactive optimization system.

6 Model Extensions and Limitations

The results outlined in the preceding section indicate that it would be quite feasible to extend the above model to more complex products that are similar to microwave modules, but

involve multiple assemblies. The modeling framework itself can be extended to accommodate additional considerations, both of the engineering and of the strategic kind, via the introduction of appropriate metrics. In this section we will briefly discuss some such extensions.

6.1 Modeling Quantity Discounts

One extension that would make sense for complex products (especially in a high-volume setting) would be to explicitly model the quantity discounts associated with placing more orders with the same supplier. A simple approach would be as follows: we assume that we receive a ‘payback’ (in dollars) when we place additional orders with a supplier; the payback will capture quantity discounts and other intangible benefits, such as better (on-time) performance, etc. The payback can be assumed to depend on the amount of business that we award to the supplier, and would increase for a while and then flatten out. It would be a piecewise linear function, $d(u)$, where u stands for the dollar value of the business awarded to the supplier. It is possible to incorporate such pay-back structures into the IP formulation described in Section 5.1 by defining appropriate indicator variables that determine the volume of business awarded to each supplier. However, the resulting formulation would almost certainly not be as strong as that in Section 5.1, and would thus call for additional analysis.

6.2 Modeling the Effect of Supplier Contracts

A more interesting effect to model would be that of business volume on the quality (defect rates) of the components supplied by a vendor. Typically, firms enter into contracts with their suppliers; these contracts *require* the supplier to meet certain quality standards specified in the contract—failure to do so results in the supplier paying a penalty to the purchaser (see, for instance, Reyniers and Tapiero, 1995). Suppliers are usually willing to commit to higher quality standards only if they are assured of a certain volume of business. This effect can be incorporated into our model by replacing the component defect rate, α_j , in the yield expression, (2), by $s_i\alpha_j$, where s_i is a supplier specific multiplier ($0 \leq s_i \leq 1$)

that depends on the volume of business that is awarded to the supplier—if the supplier is awarded the maximum possible business, then $s_i = 1$, and the component would suffer only from its *inherent* defect rate, α_j ; lower business volumes would lead to a value of $s_i < 1$, and the component’s defect rate would now worsen, reflecting the fact that the supplier lacks sufficient incentive to guarantee the maximum possible quality. Of course, such a redefinition of component defect rates will lead to additional indicator variables and constraints, as in the preceding section.

6.3 Incorporating Demand Considerations

Thus far, our quality considerations have been restricted to manufacturing yield. A far more important metric, from a customer’s point of view (and from a business strategy viewpoint), is *product quality*. However, quality, by itself, is a very difficult term to define, and to quantify. Consequently, it might be best to approach it indirectly, via demand considerations. We could begin by examining the characteristics that a product should possess, if it is to succeed in the market (i.e., the features that are “in demand”)—these would serve as surrogates for the determinants of product quality. For instance, in the case of an automobile, customers typically value safety, reliability, fuel efficiency, comfort, and appearance (not necessarily in that order). The principal observation that we make concerning such features is that they exhibit *complementarities* (i.e., reinforce one another), and that the marginal return to additional features (in terms of increased demand for the product) is likely to be diminishing. This suggests that it might be possible to model demand by means of a *supermodular* function. Given such a function, the next step would be to tie it in with the model that we have at hand for the product. Intuition suggests that the best way to do this would be via the suppliers. For instance, there might be brand name issues to consider; a computer maker might prefer an Intel microprocessor (even though cheaper/equivalent alternatives exist), simply because the “Intel Inside” stamp might increase sales considerably.

6.4 Modeling Product Families

The model that we described in Sections 4 and 5 could be extended easily to model an entire product family - each product in the family would be represented by an AND/OR tree. These otherwise independent AND/OR trees would be linked, of course, via the suppliers. The first step would be to simplify the AND/OR trees corresponding to the individual products, in order to prevent the model from becoming intractably large. This is most easily done by eliminating all components/assemblies that are not deemed to be 'important enough'. Next, the issues that we discussed in Sections 6.1, 6.2, and 6.3 would need to be explicitly considered. We might also want to consider other supplier related issues, primarily logistics costs (see Thomas and Griffin 1996), and the compatibility of the suppliers with each other. Not much is known about the latter at this stage. Some research has been reported on the strategic classification of manufacturing firms (see, for instance, Miller and Roth 1994); however, we are not aware of any quantitative models.

6.5 Conclusions

This paper has presented a formal framework for product design, that permits the consideration of engineering and strategic objectives at the design stage. The model is general enough to accommodate different application settings, and results in mathematical formulations that are strong enough to permit incorporation into interactive decision support systems. Future work will examine extensions of this work in the directions outlined above. Another planned line of inquiry is the investigation of the tradeoffs between the objectives—Section 4.3 showed how the tradeoff curve contains information pertaining to the marginal rate of substitution between the two objectives considered in that application. It would be useful to have similar information in the multiobjective case.

Appendix: Proof of Theorem 1

We make use of the following theorem (found on page 542 of Nemhauser and Wolsey 1988):

Theorem 3 *The following statements are equivalent.*

1. *An $m \times n$ integral matrix A is TU.*
2. *For every $J \subseteq N = 1, \dots, n$, there exists a partition J_1, J_2 of J such that*

$$\left| \sum_{j \in J_1} a_{ij} - \sum_{j \in J_2} a_{ij} \right| \leq 1 \quad \text{for } i = 1, \dots, m.$$

We now show that the columns of the constraint matrix associated with an AND/OR tree can be partitioned in a manner consistent with the requirements of Theorem 3. It is sufficient to prove the result for $J = N$, since taking a subset of N corresponds to deleting certain variables/tree nodes, which in turn spawns several independent AND/OR trees. First, note that any pair of nodes can appear together in at most one constraint. Second, the only nodes that can occur together in a constraint are those which share a parent-child relationship. Now we define two rules which determine how we partition the variables into two disjoint sets. First, consider a generic AND-constraint (Rule 1): $x - y = 0$. Clearly, when partitioning the columns of the constraint matrix, x and y will have to belong to the same set, in order to satisfy the conditions set forth in the theorem. Next, consider a generic OR-constraint (Rule 2): $-x + y_1 + y_2 + \dots + y_k = 0$. If k is an even number, we can let $x, y_1, \dots, y_{k/2}$ belong to one set and $y_{k/2+1}, \dots, y_k$ belong to the other set. Similarly, if k is odd, $(x, y_1, \dots, y_{\lfloor k/2 \rfloor + 1})$ and $(y_{\lfloor k/2 \rfloor + 2}, \dots, y_k)$ would be the two sets.

Given an AND/OR tree, we can write the constraints in a systematic, *breath-first* manner. Now, starting at the first constraint, we partition the variables into two sets, using the two rules given above. The fact that we generated the constraints in a breath-first fashion, allied with our earlier observations, ensures that each time a constraint is encountered, there will be exactly one variable (in that constraint) which has already been assigned to a set; the other variable(s) in the constraint can be appropriately assigned depending on whether it is an AND-constraint or an OR-constraint. Thus, there will never be a conflict, i.e., we will never encounter a variable which cannot be assigned to either set without violating

the requirements of the theorem. At the end of this process, the variables will have been partitioned into two sets satisfying the conditions of Theorem 3. \square

References

- [1] Arntzen, B.C., Brown, G.G., Harrison, T.P., and Trafton, L.L. 1995. Global Supply Chain Management at Digital Equipment Corporation. *Interfaces*, **25**, 1995, 69–83.
- [2] Bakerjian, R., editor. *Design for Manufacturability*, volume 6 of *Tool and Manuf. Engineers Handbook*. SME, 1992.
- [3] Ball, M. O., Baras, J. S., Bashyam, S., Karne, R. K., and Trichur, V. 1995. On the selection of parts and processes during design of printed circuit board assemblies. In *Proc. of the INRIA/IEEE Symp. on Emerging Technologies and Factory Automation*, vol. 3, 241–249. IEEE Computer Society Press, Los Alamitos, CA.
- [4] Boothroyd, G. and Dewhurst, P. 1983. *Design for Assembly—A Designer’s Handbook*. Dept. of Mech. Eng., Univ. of Massachusetts at Amherst, 1983.
- [5] Bralla, J., editor. 1986. *Handbook of Product Design for Manufacturing*. McGraw Hill, 1986.
- [6] Brayton, R.K. and Spence, R. 1980. *Sensitivity and Optimization*. Elsevier Scientific Publishing Company, 1980.
- [7] Brown, C.G., Graves, G.W., and Honczarenko, M.D. 1987. Design and operation of a multicommodity production/distribution system using primal goal decomposition. *Management Science*, **33**, 1987, 1469–1480.
- [8] Cohen, M.A., and Lee, H.L. 1989. Resource deployment analysis of global manufacturing and distribution networks. *Journal of Manufacturing and Operations Management*, **2**, 1989, 81–104.

- [9] Dyer, J. H. 1990 Specialized Supplier Networks As A Source Of Competitive *Strategic Management Journal*, **17**, 1990, 271–291.
- [10] Feldmann, K. and Franke, J. 1993. Computer-Aided Planning Systems for Integrated Electronic and Mechanical Design. *IEEE Trans. on Comp., Hybrids, and Manuf. Tech.*, **16**, 1993, 377–383.
- [11] Gass, S.I. 1985. *Linear Programming*, Fifth Edition. International Thomson Publishing, Inc., 1985.
- [12] Goicoechea, A., Duckstein, L., and Zionts, S., editors. *Multiple Criteria Decision Making: Proceedings of Ninth International Conference*. Springer-Verlag, 1992.
- [13] Gupta, S. K. 1994. *Automated Manufacturability Analysis of Machined Parts*. Ph.D dissertation, University of Maryland, College Park.
- [14] Harhalakis, G.; Minis, I.; and Rathbun, H. Manufacturability Evaluation of Electronic Products Using Group Technology. In *Proceedings of the 1993 NSF Design and Manufacturing Systems Conference*, pages 1353–1360, Charlotte, NC, 1993.
- [15] Heng, C. A. S. and Gay, R. K. L. 1991 “Design For Manufacturability: Cost Analysis of Electronic Printed Circuit Board Assembly”. In G.A. Gabriele, editor, *Advances in Design Automation - Volume 1*, pages 63–67. ASME, 345 East 47th Street, United Engineering Center, New York, NY 10017, 1991.
- [16] Horowitz, E., Sahni, S., and Rajasekaran, S. 1998. Computer Algorithms. W.H. Freeman and Co., 1988.
- [17] Jakiela, M. and Papalambros, P. 1989. Design and implementation of a prototype intelligent CAD system. *ASME J. of Mech., Transm., and Autom. in Design*, **111**, 1989, 252–258.

- [18] Miller, J. G.; and Roth, A. V. 1994. A Taxonomy of Manufacturing Strategies *Management Science*, **40**, 1994, 285–303.
- [19] Nemhauser, G. L. and Wolsey, L. A. 1988. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.
- [20] Oh, C.J. and Park, C.S. 1993 “An Economic Evaluation Model for Product Design Decisions under Concurrent Engineering”. *The Engineering Economist*, **38**, 1993, 275–297.
- [21] Philip, J. 1972. Algorithms for the Vector Maximization Problem. *Mathematical Programming*, **2**, 1972, 207–229.
- [22] Preparata, F.P., and Shamos, M.I. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
- [23] Reyniers, D. J.; and Tapiero, C. S. 1995. The Delivery and Control of Quality in Supplier-Producer Contracts. *Management Science*, **41**, 1995, 1581–1589.
- [24] Russell, G. A. “Design for Assembly of Printed Circuit Boards”. In *Proceedings of the First International Conference on Product Design for Assembly*, Newport, RI, April 15-17 1986.
- [25] Serafini, P., editor. *Mathematics of Multiobjective Optimization*. Springer-Verlag, 1985.
- [26] Splain, J. “A User Interface for Discrete Optimization Based Tradeoff Analysis”. MSSE Thesis, University of Maryland, 1998.
- [27] Thomas, D. J.; and Griffin, P. M. 1996. Coordinated Supply Chain Management *European Journal of Operational Research*, Vol. **94**, 1996, 1–15.
- [28] Tits, A. L.; Ma, Z. 1986. Interaction, Specification Refinement, and Tradeoff Exploration in Optimization-Based Design of Engineering Systems. In *Proc. of the 1985 IFAC*

Workshop on Control Applications of Nonlinear Programming and Optimization, 189–194. Pergammon Press.

- [29] Trichur, V. S.; Ball, M. O.; Baras, J. S.; Hebbar, K.; Minis, I.; Nau, D.; and Smith, S. J. J. 1996. Integrating Tradeoff Analysis And Plan-Based Evaluation Of Designs For Microwave Modules. In *Proceedings of the Agile and Intelligent Manufacturing Symposium*, Rensselaer Polytechnic Institute, NY, 1996.