Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1999), *13*, 27–36. Printed in the USA.
Copyright © 1999 Cambridge University Press 0890-0604/99 $12.50

On utility of inductive learning in multiobjective robust design

BABAK FOROURAGHI
Computer Science Department, Saint Joseph's University, Philadelphia, PA 19131

(Received December 1, 1997; Accepted October 26, 1998)

Abstract

Most engineering design problems involve optimizing a number of often conflicting performance measures in the presence of multiple constraints. Traditional vector optimization techniques approach these problems by generating a set of Pareto-optimal solutions, where any specific objective can be further improved only at the cost of degrading one or more other objectives. The solutions obtained in this manner, however, are only single points within the space of all possible Pareto-optimal solutions and generally do not indicate to designers how small deviations from predicted design parameters settings affect the performance of the product or the process under study.

In this paper we introduce a new approach to robust design based on the concept of inductive learning with regression trees. Given a set of training examples relating to a multiobjective design problem, we demonstrate how a multivariate regression tree can utilize an information-theoretic measure of covariance complexity to capture optimal, tradeoff design surfaces. The novelty of generating design surfaces as opposed to traditional points in the design space is that now designers are able to easily determine how the responses of a product or process vary as design parameters change. This ability is of paramount importance in situations where design parameter settings need to be modified during the lifetime of a product/process due to various economic or operational constraints. As a result, designers will be able to select optimal ranges for design parameters such that the product's performance indices exhibit minimal or tolerable deviations from their target values. To highlight the advantages of our methodology, we present a multiobjective example that deals with optimum design of an electric discharge machining (EDM) process.

Keywords: Pareto-optimality; Regression Trees; Robust Design; Entropy; Inductive Learning

1. INTRODUCTION

Most engineering design problems involve optimizing several often noncommensurate performance indices (objectives). The problem of finding an optimum vector-valued objective is commonly referred to as *vector optimization* in the literature, and solutions of such problem are called *noninferior* or *Pareto-optimal*. Determination of Pareto-optimal solutions in a given optimization task requires finding a vector of optimal objectives where an individual objective can be further improved only at the cost of degrading at least one other objective (Hwang & Masud, 1979; Chankong & Haimes, 1983). Typically, vector optimization utilizes mathematical programming, goal programming, utility theory, etc. (Hwang & Masud, 1979) or statistical approach of multivariate analysis of variance (MANOVA), which is used extensively in the area of quality control (Harris, 1985). Regardless of their underlying methodological differences, these techniques generate Pareto-optimal solutions that lack two crucial characteristics: First, these solutions are represented as "points" in both the space of design variables (independent parameters) and objectives (dependent parameters); and second, the obtained solutions are rigid in that they do not provide any understanding of the complex nature of the underlying problem that is to be solved.

To clarify, in many real-world situations, either due to processing limitations or economic factors, it is nearly impossible to pinpoint a singular point as an optimum design vector. For example, setting a beam's diameter at 3.3 mm may not be possible either due to machine processing limitations or presence of high degrees of variability in the man-

Reprint requests to: Babak Forouraghi, Computer Science Department, Saint Joseph's University, Philadelphia, PA 19131, U.S.A. Telephone: (610) 660-1167; E-mail: bforoura@sju.edu

https://doi.org/10.1017/S0890060499131032 Published online by Cambridge University Press

ufacturing process (e.g., 3.3 ± 0.1 mm). To deal with such variations, designers need to consider two factors: 1) how unexpected deviations from an optimum design degrade the overall performance of a product, and 2) how the performance of the product under study can become least sensitive to such deviations. These considerations are the focal point of the field of robust design (Phadke, 1989).

Importantly, in terms of obtaining an understanding of the nature of the problem at hand, traditional mathematical and statistical techniques only identify the quantitative input-output behavior of a system that is to be optimized. In other words, a conventional optimizer recommends to designers a Pareto-optimal design vector without conveying what actually constitutes the optimality of the generated solution. If designers need to get a better understanding of the underlying optimization knowledge, they need to iterate a particular optimizer several times to determine how deviations from the recommended design affect the overall product performance indices.

Having stated the two major disadvantages of the traditional approaches to multiobjective optimization, we now discuss what kinds of tools are needed to potentially remedy the situation. The actual learning of the optimization knowledge in a given task is crucial in that not only it allows determination of optimal settings of design variables, it also allows systematic examination of alternative design scenarios. The learning process, which can be defined as acquisition, assimilation and restructuring of knowledge, has received a great deal of attention from researchers in the field of artificial intelligence (AI) (Shavlik & Dietterich, 1990). For example, symbolic search techniques based on the MOA* algorithm (Bradley & White, 1991), although limited in their applicability to real-world problems, were developed as multiobjective generalizations of the heuristic search algorithm A*.

Another viable AI approach to multiobjective optimization is machine learning, or more specifically, inductive learning (Carbonell et al., 1987). Assuming that the input-output components of a system (i.e., a product or a process) can be described as a set of attribute-value design vectors, an inductive learner can inspect this set and discover highly complex relationships between system inputs and outputs and represent them in forms that can easily be examined by designers. Decision-tree classifiers (Quinlan, 1986), classification and regression trees (CART) (Breiman et al., 1984), and inductive partitioning with regression trees (IPRT) (Shien & Joseph, 1992) are some of the more widely used inductive learning systems. The common operational characteristic of all these methods is that they view induction as a form of optimization (Rendell, 1990).

By nature, decision trees are best suited for classification purposes and cannot be directly applied to regression. Regression trees, on the other hand, can manipulate the real-valued continuum of responses and are more suitable for knowledge discovery in multiobjective optimization. The major limitation of regression-tree algorithms, such as CART or IPRT, is that they are designed for univariate regression and cannot easily handle multiobjective optimization, which requires simultaneous optimization of several responses.

In this paper, we present a new framework within which multiobjective optimization is accomplished through induction of multivariate regression trees. Particularly, we present a tree partitioning algorithm that utilizes an information-theoretic measure of covariance complexity (Bozdogan, 1990) to transform highly convoluted regression surfaces into a number of simpler and smaller subsurfaces. It will be demonstrated that these subsurfaces are minimal volume hyper-ellipsoidal (or simply, ellipsoidal) regions in the feasible space of objectives and constraints with three important properties: First, they are mapped from hyper-rectangular design regions and not singular design points; second, the degree of interaction among system responses in a given ellipsoid is minimal; and third, an *a posteriori* analysis of the discovered regions allows designers to select noninferior solutions that satisfy their preferences.

2. MULTIVARIATE REGRESSION TREES

The basic element for inducing a multivariate regression tree is a set of training examples that provides a capsule view into the relationship between design variables (independent parameters) and objectives/constraints (dependent parameters). The inductive learner uses these examples as a source of knowledge and incrementally decomposes a complex regression surface into a number of simpler regression subsurfaces. This piecewise model decomposition is accomplished by successive partitioning of the training population at each level of the tree in an attempt to identify minimal-volume optimal clusters in the response region. The minimal-volume requirement for a product's response is important in that it guarantees that design variations due to operational or manufacturing limitations will not drastically deteriorate product performance. The following provides more details regarding the induction process.

Given a learning sample $L = \{(X_1,Y_1),\cdots,(X_k,Y_k)\}$, where each training example (X_i,Y_i) associates a design vector $X_i = (x_{i1},\cdots,x_{in})$ with a specific response vector $Y_i = (y_{i1},\cdots,y_{ip})$, the learning algorithm generates a prediction rule $d{:}R^n \to R^p$ that is a mapping from the n-dimensional space of design variables (attributes) to the p-variate objectives/constraints space.

Initially, all k training examples, denoted by population $N(\mu,\Sigma)$ with mean vector μ and covariance matrix Σ, reside at the root of an empty tree. Following a divide-and-conquer approach, the root node is partitioned into two left and right nodes, such that k_1 of the original k examples fall in the left node and the remaining k_2 cases fall in the right node, hence $k = k_1 + k_2$ (see Fig. 1). The splitting of a parent into two offspring nodes is accomplished by selecting an attribute and a threshold for partitioning the attribute's range into two regions (Fayyad & Irani, 1992). Among all possible attribute/threshold pairs, the pair that results in the

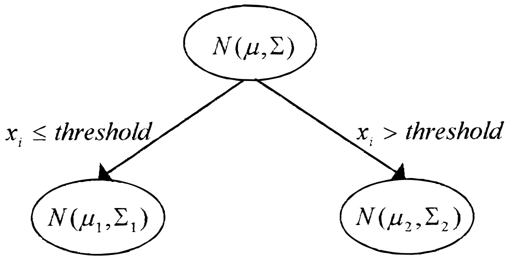

Fig. 1. Splitting of a tree node.

"best" split, where the resulting left and right nodes maximize some measure of fitness, is selected and the node is split accordingly.

The process of partitioning is then recursively applied to all newly generated nodes until some stopping criterion is met. In our case, to ensure nonsingularity of Σ, a multivariate heuristic that dictates that the number of examples in a node has to be at least as large as the number of responses was used (Gnanadesikan, 1977). Further, after a tree is completely grown in the prescribed manner, some type of pruning will prove beneficial should the problem of overspecialization cause detrimental effects on overall efficiency of the learning system (Quinlan, 1986).

Upon completion of the learning phase, the induced tree contains a number of paths, each starting from the root and ending in a terminal node or leaf. Each path pinpoints a regression subsurface by the virtue of examples that are contained in its leaf. These examples define an ellipsoidal cluster E in the p-variate response region:

$$E = \{y \in R^p \,|\, (y-\mu)'\Sigma^{-1}(y-\mu) \ge 0\}, \tag{1}$$

where μ is the center of the ellipsoid, Σ is the ellipsoid matrix, which is positive and semidefinite, and R^p is the Euclidean space of p dimensions. The goodness of these clusters is then determined by a partitioning technique that is based on Bozdogan's concept of multivariate entropy.

Note that the node-attribute tests along a particular path of an induced tree act as intersecting hyperplanes that together define a hyper-rectangular region in the feasible design space. Our primary objective is to map minimal entropy ellipsoids in objectives space from such hyper-rectangles. Figure 2 illustrates an example where the feasible design region is identified by geometric constraints g_1, g_2, and g_3. The inductive learner, denoted by the predictor $d(X)$, maps an identified rectangular region to an optimal elliptical region in the objectives space.

2.1. Covariance complexity

Bozdogan's information-theoretic measure of covariance complexity is typically used for selection and evaluation of multivariate models (Bozdogan, 1990). Basically, the covariance complexity metric measures how the individual subcomponents of a model or a system interact with one another. In the case of multivariate regression trees, we use a tree as a representative of an underlying model that is to be captured through the induction process. The main underlying assumption in our approach is that a continuous, multivariate joint density function $f(Y)$ with a mean vector $\mu = (\mu_1,\cdots,\mu_p)$ and a $p\times p$ positive, semidefinite covariance matrix Σ can be defined as:

$$f(Y) = f(y_1,\ldots,y_p) = (2\pi)^{-0.5p}|\Sigma|^{-0.5}e^{-0.5(Y-\mu)'\Sigma^{-1}(Y-\mu)}, \tag{2}$$

where $Y \sim N_p(\mu,\Sigma)$.

The *marginal entropy* $H(y_j)$ and *joint* entropy $H(Y)$, can then be expressed using Eqs. (3) and (4) as follows:

$$H(y_j) = -E[\log(f(y_j))] = -\int_{-\infty}^{\infty} f(y_j)\log(f(y_j))dy_j. \tag{3}$$

$$\begin{aligned} H(Y) &= -E[\log f(Y)] \\ &= -\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f(y_1,\ldots,y_p)\log(f(y_1,\cdots,y_p))dy_1, \\ &\quad \cdots, dy_p. \end{aligned} \tag{4}$$

As regards the information-theoretic measure of complexity, Bozdogan defines the quantity $I(Y)$ as the measure of interaction between variables $y_1,\cdots,y_p$ and expresses it in terms of the previously defined marginal and joint entropies:

$$I(Y) = I(y_1,\cdots,y_p) = \sum_{j=1}^{p} H(y_j) - H(y_1,\cdots,y_p). \tag{5}$$

Given the previous definition of joint entropy, $H(Y)$ is computed as:

$$H(Y) = 0.5p\log(2\pi) + 0.5\log|\Sigma| + 0.5E[(Y-\mu)'\Sigma^{-1}(Y-\mu)]. \tag{6}$$

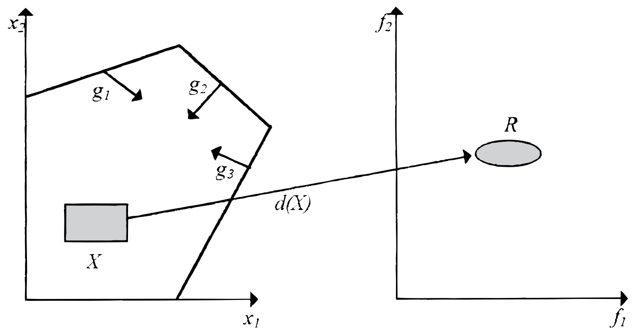

Fig. 2. Mapping of feasible subregions in R^n to optimal clusters in $R^p (n = 2, p = 2)$.

And by observing that the expected value of the quadratic form in Eq. (6) follows a Chi-squared distribution with p degrees of freedom:

$$E[(Y-\mu)'\Sigma^{-1}(Y-\mu)] = E[\chi_p^2] = p \tag{7}$$

we easily obtain the following two expressions for the joint and marginal entropies, $H(Y)$ and $H(y_j)$, respectively $(j-1,\cdots,p)$.

$$H(Y) = 0.5p\log(2\pi) + 0.5\log|\Sigma| + 0.5p \tag{8}$$

$$H(y_j) = 0.5\log(2\pi) + 0.5\log(\sigma_j^2) + 0.5 \tag{9}$$

The total amount of interaction or *informational complexity* $I(Y)$ is defined as:

$$\begin{aligned} C_o(\Sigma) &= I(y_1,\cdots,y_p) \\ &= \sum_{j=1}^{p}[\log(2\pi) + 0.5\log(\sigma_j^2) + 0.5] - 0.5\log(2\pi) \\ &\quad - 0.5\log|\Sigma| - 0.5p, \end{aligned} \tag{10}$$

which in turn reduces to:

$$0.5\sum_{j=1}^{p}[\log(\sigma_j^2)] - 0.5\log|\Sigma| = 0.5\log\prod_{j=1}^{p}(\sigma_j^2) - 0.5\log|\Sigma|. \tag{11}$$

But because the geometric mean of the individual variances $\sigma_1^2,\cdots,\sigma_p^2$ can be manipulated as:

$$\frac{p}{2}\log\left[\prod_{j=1}^{p}\sigma_j^2\right]^{1/p} \tag{12}$$

and also because of the following inequality relation between the arithmetic and geometric means of the individual variances:

$$\frac{1}{p}\sum_{j=1}^{p}\sigma_j^2 \geq \left[\prod_{j=1}^{p}\sigma_j^2\right]^{1/p} \tag{13}$$

we can maximize the total amount of information gain $I(Y)$ by rewriting $C_o(\Sigma)$ as:

$$\frac{p}{2}\log\left[\frac{1}{p}\sum_{j=1}^{p}\sigma_j^2\right] - 0.5\log|\Sigma|. \tag{14}$$

The sum of variances is Eq. (14) is the *trace* (tr) of the covariance matrix $\mathbf{\Sigma}$, so Bozdogan's final expression for the measure of covariance complexity becomes:

$$C_o(\Sigma) = \frac{p}{2}\log\left[\frac{tr(\Sigma)}{p}\right] - \frac{1}{2}\log|\Sigma|. \tag{15}$$

At each level of partitioning then a given node's population of responses is divided into two subpopulations such that the covariance complexity of each of the two resulting subpopulations is minimal.

It is implied from examining Eq. (15) that trace and determinant of dispersion matrix play key roles in minimization of C_o: Minimization of trace results in clusters where individual variances are as small as possible; and minimization of the determinant helps locate regions where interaction among responses is minimal. The determinant of the dispersion matrix has another important characteristic, which will be discussed in the next section.

2.2. Optimal response clusters

In Section 2.1 we showed that our partitioning algorithm is biased toward minimal entropy clusters. To satisfy robust design and multiobjective optimization requirements, we need to verify two additional properties for generated ellipsoids: minimal volume criterion and Pareto optimality.

In terms of the minimal volume requirement, further examination of Eq. (15) reveals that minimization of $C_o(\Sigma)$ during the tree induction process also identifies clusters that have minimal *generalized variance*, $|\Sigma|$. Assuming that a cluster's response vectors form a parallelotope, as it is a typical view in the case of N-space representation of multivariate observations, the generalized variance is defined as the square of the p-dimensional "volume" of such parallelotopes (Tatsuka, 1971). Minimization of $C_o(\Sigma)$, therefore, results in identification of minimal volume response clusters where a product is least sensitive to design perturbations.

To address Pareto-optimality of clusters, we first assume, without loss of generality, the general problem of simultaneously minimizing p responses. The following definitions are then adopted (Hwang & Masud, 1979):

Definition 1. A vector $u = (u_1,\cdots,u_p)$ is said to be *inferior to* (*dominated by*) vector $\boldsymbol{v} = (v_1,\cdots,v_p)$ if and only if $\forall i \in \{1,\cdots,p\}$, $v_i \leq u_i \wedge \exists i \in \{1,\cdots,p\}|v_i < u_i$.

Definition 2. Vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ *are said to be noninferior* to each other if neither $\boldsymbol{u}$ is inferior to $\boldsymbol{v}$ *nor* $\boldsymbol{v}$ *is inferior to* $\boldsymbol{u}$.

As an example of noninferiority, consider Fig. 3. Here, vectors $\boldsymbol{u} = (f_1^u, f_2^u)$ and $\boldsymbol{v} = (f_1^v, f_2^v)$, which lie on the boundary of the feasible region (Pareto front F), are noninferior to each other because $f_1^v > f_1^u$, but $f_2^v < f_2^u$.

Definitions 1 and 2 are suitable for singular response vectors, but they need to be extended to discuss noninferiority of ellipsoidal regions. Thus, we define two clusters E_1 and E_2 to be *noninferior* if their centers μ_1 and μ_2 are noninferior to each other in the sense of Definition 2, and that they contain portions of the Pareto front that may or may not be overlapping. Given the fact that there are infinitely many Pareto solutions, the centers of clusters E_1 and E_2 may be far enough from each other such that each cluster contains

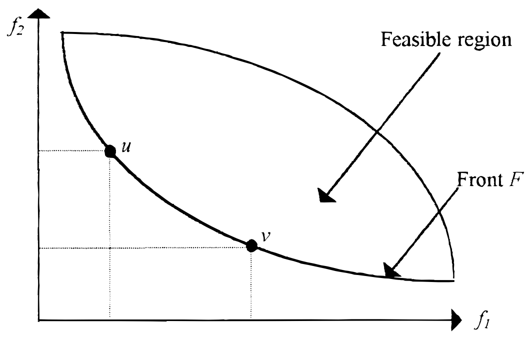

Fig. 3. Tradeoff vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.

distinct segments of the Pareto front. If cluster centers are too close, however, they may contain overlapping sections of the Pareto front.

The notion of noninferiority of clusters, as discussed above, is only the first step in approaching a multiobjective optimization problem. To select a set of candidate noninferior solutions and negotiate tradeoffs between them, a decision process also needs to be developed.

In general, depending on how the computation of tradeoff solutions is conducted, multiobjective optimization can be categorized as *a priori*, *interactive* (*progressive*), or *a posteriori* (Chankong & Haimes, 1983). In the first category, decision-maker's preferences are first built into the system and then the search for the Pareto frontier proceeds; in the second category, decision-making and optimization are interleaved; and finally, in the third category, an optimizer produces a set of candidate noninferior solutions before designers express any preferences. Our approach combines elements from *a priori* and *a posteriori* articulation techniques as follows: In its *a priori* phase, the induction process progressively attempts to locate minimal volume clusters in the response region as discussed before. It is conceivable that not all of the generated clusters conform to designers' preferences. Therefore, during the *a posteriori* phase designers indicate their preferences in the form of desired ranges for product responses, and the algorithm automatically searches a generated tree for noninferior ellipsoids whose range of responses are within user-defined margins. An added benefit of this type of search is that it serves as an inverse mapping by which designers can determine how a product's geometry changes as various performance regions in the objectives space are considered.

3. RELATED WORK

In this section we discuss three approaches that in principal are related to our methodology for region approximation: First, ellipsoidal design centering; second, concurrent subspace optimization; and third, the general framework of response surface methodology.

3.1. Ellipsoidal design centering

The ellipsoidal method for feasible region approximation is a numeric approach that, based on assumptions of multinormal distribution of its random variables and convexity of the feasible region, generates a sequence of ellipsoids of decreasing volume. (Abdel-Malek & Hassan, 1991). These ellipsoids have the properties that they are bounded within the feasible region and that they all converge to a single ellipsoid whose center is the proposed design center. Computation of the final ellipsoid is facilitated by a linear search that essentially relies on a single quadratic equation to approximate the boundary of a region defined by several nonlinear constraints. Obtained design ellipsoids in this manner can be viewed as nominal parameter values that maximize either the tolerances on parameters or the yield for an assumed distribution of parameters around their nominal values.

Our methodology is different from design centering in the sense that we conduct the search for minimal volume ellipsoids in the p-variate space of objectives/constraints and not in the space of design parameters. Other major differences between the two techniques are: 1) our technique does not operate under the assumption of feasible region convexity; 2) multivariate induction process discovers hyperrectangular design regions in R^n, which identify ellipsoids in the response surface R^p; and 3) the regression-tree method is symbolic and allows designers to easily examine a generated tree and discover how changing a design parameter setting affects the behavior of a product.

3.2. Concurrent subspace optimization

The second related work, concurrent subspace optimization (CSSO), is a multidisciplinary design optimization (MDO) technique (Lokanathan et al., 1995) in which complex designs are decomposed into subsystems that can be optimized independently and concurrently (Lokanathan et al., 1996). Specifically, in the area of integrated circuit fabrication, coordination of process and circuit design activities is a challenging problem that often requires satisfying conflicting objectives. CSSO allows circuit and process subspace optimization modules to proceed independently while attempting to optimize some global function. This type of decomposition, however, requires that a subsystem be able to approximate how design changes made in its own domain will affect the value of a global objective. These local approximations are handled by solving global sensitivity equations that rely on gradient information of system outputs with respect to system inputs. Each subspace then stores a number of design points in a central database, generally in the direction that improves local subspace performance. At the end of subspace optimization, a coordination mechanism constructs a response surface using the visited design points by each subspace, negotiates tradeoffs, and performs global optimization using the set of all design variables.

The similarity between the method of CSSO and ours is that they both seek to find estimates of optimum conditions by decomposing highly nonlinear response surfaces into a number of simpler subsurfaces. The differences between the two methods, however, are numerous:

1. computational complexity of recursive partitioning with multivariate regression trees, similar to that of decision tree classifiers (Quinlan, 1986), is a function of dimensions of the training set (i.e., number of training examples in a tree node and number of design variables) and does not rely on computation of numerically expensive gradient information;
2. the visited set of design points in our approach is static and does not change as optimization progresses;
3. we map feasible regions in objectives space from feasible regions in design variables space;
4. the regression tree's decomposition of a response surface relies on statistical properties of a product's performance indices and is performed automatically without any user intervention; and
5. CSSO is inherently a multidisciplinary, distributed optimization technique.

So, our divide-and-conquer approach could potentially decompose any response surface into as many subsurfaces as possible without the need to distinguish among the involved optimization disciplines. To avoid repetition here, we refer the reader to Section 2.0 to verify this last point.

3.3. Response surface methodology (RSM)

Generally, utilization of RSM within the framework of univariate product optimization assumes that a response η is a function of a set of design variables $x_1, \cdots, x_n$ and that the function can be approximated in some region of the x's using a first-order polynomial model:

$$\eta = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{16}$$

or a second-order model:

$$\eta = \beta_0 + \sum_i \beta_i x_i + \sum_i \beta_{ii} x_i^2 + \sum_{i<j} \sum_j \beta_{ij} x_i x_j \tag{17}$$

should there be a strong evidence of curvilinearity or interaction among system inputs and output (Myers et al., 1989). Note that poor model parameters in RSM result in low correlation coefficient R^2.

After a series of statistically balanced experiments (Phadke, 1989) has been conducted and design sample points have been collected, model approximation is carried out using ordinary least squares for estimation of the coefficients β of the regression model. Upon discovering a "fit" model using multiple regression, the method of steepest ascent for sequentially moving toward the optimum response follows. The obtained surface is then used in three ways: 1) to describe how design variables affect the response; 2) to determine the interrelationships among the design variables; and 3) to describe combined effect of all design variables on the response (Giovanni, 1983).

Unfortunately, multiresponse surface analysis is not as developed as its univariate counterpart (Myers et al., 1989). In principle, an RSM-based optimizer uses fractional factorial experiments, thus requiring far fewer samplings of the design and objectives spaces than does our approach. In terms of computational complexity, the regression-tree's symbolic-model-building approach is numerically less intensive than the method of least squares, which is used for estimation of regression coefficients (Murthy, 1995).

4. A DESIGN EXAMPLE

In this section, we present a canonical example relating to optimization of an electric-discharge-machining (EDM) process (Osyczka, 1984). Before discussing the actual results, however, three crucial issues that are directly related to performance evaluation of a learning system must be closely examined: 1) preparation of training and testing cases, 2) variance stabilization for our parametric partitioning rule, which relies exclusively on at least the assumption of normal distribution of the underlying multivariate observations, and 3) the regression error analysis.

4.1. Resampling techniques

An important step toward construction and evaluation of a learning system is the preparation of training and testing samples. An obvious solution is to perform training and testing on the same set of data and compute the *apparent error rate*, which can be a highly overoptimistic estimate of a learning system's performance. A statistically rigorous technique for producing more accurate error estimates is *random subsampling*, where a data set is randomly divided into independent training/testing samples, and the overall error estimate is computed as the average of the error rates obtained from independent test cases (Weiss & Kulikowski, 1991). Methods such as k-fold cross-validation are particularly useful in situations where the training and testing samples are scarce, and a moderately sized sample must serve as the training and testing sets. For example, in many real-world applications, designers have to rely on finite element analysis to assess various design scenarios. Due to numeric-intensive nature of such analytical operations, only smaller sets of training examples can be produced over reasonably tolerable amounts of time.

In the EDMP design example the system responses were known in closed form, thus we were able to produce any number of training and testing cases. Given two design variables

x_1 and x_2 subject to certain constraints (see Section 4.4), we uniformly sampled the design region (x_1, x_2) in 900 points and obtained the needed training/testing vectors. This sample was then randomly shuffled and divided into the two sets L_{200} and T_{700} of size 200 and 700, respectively. The set L_{200} was used in its entirety for generating regression trees, and regression errors were computed using the set T_{700}. This entire process of random selection of training samples, learning, and testing was repeated a total of five times so that results could be presented with 95% confidence (t-distribution). In this particular example, empirical studies indicated training samples smaller than 200 (i.e., 200 function evaluations) were not able to convey enough information to the learning algorithm. In general, the issue of how large a sample should be to achieve certain learning accuracy is addressed by the field of computational learning theory (Shavlik & Dietterich, 1990). This issue is beyond the scope of our paper, but a relevant example (Mehrotra et al., 1991), which finds the bounds on the number of samples needed for performing neural learning, addresses the various salient issues.

4.2. Variance stabilization

When building linear models, a crucial consideration is with the effects of violations of distributional assumptions in multivariate statistical analysis (Hahn, 1971). The parametric splitting rule used in this work is derived under at least the assumption of multivariate normality, which ensures that the data vectors are independent random samples from a population in which any linear combination of the variables in the data vector is normally distributed. In reality, however, it may not be possible to satisfy the normality requirement due to the skewed distributional nature of the gathered data. A common solution for this situation is to use one of two types of variance stabilization techniques: *log* transformation and *rank* transformation.

The log transformation is a powerful technique that smoothes the underlying data distribution in such a way that the log of a non-normal response is more likely to follow or approach normal distribution, while the original response will then have a lognormal distribution (Hahn, 1971). Further, aside from elimination of some of the nonlinear components of the original responses, the log-transformed data causes the arithmetic means to be transformed into geometric means, which are better indicators of central tendencies of a population. Log transformation are routinely used in the field of optimization especially in the area of quality control (Phadke, 1989).

The rank-transformation procedures, on the other hand, apply usual parametric procedures to the ranks of the data instead of the data themselves (Zwick, 1985). The rank-transformed data better approximate normal distribution due to the fact that ranks are drawn from a population (natural numbers) where individual members have equal probability of occurrence. Another advantage of rank transformed data is that instead of using nonparametric tests of significance, parametric models can still be used while the underlying data are readily represented in terms of their respective ranks.

In this work, both types of power transformation, namely log and rank transformations, were attempted. The empirical results obtained indicated that the log transformed data produced far more accurate results than the rank-ordered data, presumably because of log linearization of the convoluted response surfaces. The results presented in Section 4.4 are entirely based on stabilizations achieved through log transformations.

4.3. Regression error analysis

The error analysis used for univariate regression tree models such as CART (Breiman et al., 1984) or IPRT (Shien & Joseph, 1992) is based on the correlation coefficient R^2, which measures how much of variation in a response y can be explained by a regression. Given a learning sample L consisting of vectors $(X_1, y_1), \cdots, (X_k, y_k)$, where X_i and y_i, respectively, denote a design vector and a single response, we estimate the error RE^* of the generated predictor d as:

$$RE^*(d) = 1 - R^2 = \frac{\sum_i (y_i - \hat{y}_1)^2}{\sum_i (y_i - \bar{y})^2}, \tag{18}$$

where $\bar{y}$ is the mean and $\hat{y}_i$ is the computed estimate for y_i (i.e., $\hat{y}_i = d(X_i)$).

To accommodate this type of analysis for our multivariate models, the following extension was implemented. The overall regression surface in our approach is decomposed into a number of simpler regression subsurfaces, each of which is represented by a number of training examples (leaf clusters). Let us assume that during the testing phase of a tree, a test case (X_i, Y_i) falls in the leaf l identified by the normal population $N^l(\mu_l, \Sigma_l)$. The weighted Euclidean distance between response vector Y_i and cluster N^l is called the *Mahalanobis* squared distance, and is defined as (Everitt, 1974):

$$\Delta^2(Y_i, N^l) = (Y_i - \mu_l)' \Sigma_l^{-1} (Y_i - \mu_l). \tag{19}$$

Unlike the traditional Euclidean distance measure, which is based on the assumption of equal variance among all responses, the Mahalanobis distance takes into consideration the noncommensurate nature of responses and therefore accommodates intercorrelations among the responses, as well as possible differences among their variances (Gnanadesikan, 1977). Note that the Δ^2 statistic reduces to unweighted Euclidean distance if the identity matrix I replaces Σ.

Considering Eq. (19), we compute the overall error for an induced regression tree using m test cases as follows:

$$\frac{1}{m}\sum_{i=1}^{m}\Delta(Y_i, N^l), \tag{20}$$

where N^l is the leaf in which the test case Y_i falls.

4.4. The EDM process

The example presented here deals with optimization of an electric discharge machining (EDM) process (Osyczka, 1984). In this process, the main input quantities are pulse duration (T_i), pulse interval (T_0), amplitude of the discharge current (I), erosion diameter (ϕ) and erosion depth (g). The optimization process for the four objective functions can formally be stated as:

Objectives:

Maximize metal removal rate, Q_v (mm^3/min)
Minimize electrode wear, δ (%)
Minimize machine power consumption, $N(W)$
Minimize surface roughness, R_a (mm),

where dependencies between input and output quantities were experimentally determined through a series of statistical regression analyses with the following results:

$$\begin{aligned} Q_v = {} & e^{11.744} I^{0.206+0.032\ln T_i+0.022\ln T_0+0.205\ln g} \\ & T_i^{-1.555+0.047\ln T_0+0.276\ln\phi+0.05\ln g} \\ & T_0^{-0.174\ln\phi-0.107+0.155\ln g}\,\phi^{-1.067-0.124\ln}\,g^{-0.742}. \end{aligned} \tag{21}$$

$$\begin{aligned} \delta = {} & e^{-81.509} I^{5.634-0.349\ln T_0-0.335\ln T_i+0.119\ln\phi+0.174\ln g} \\ & T_i^{3.726-0.551\ln T_0-0.334\ln\phi+0.253\ln g} \\ & T_0^{-0.207\ln\phi+13.609+0.207\ln g}\,\phi^{12.219-0.71\ln g} g^{-3.102}. \end{aligned} \tag{22}$$

$$\begin{aligned} R_a = {} & e^{20.971} I^{-1.678+0.224\ln T_i-0.027\ln T_0+0.135\ln\phi-0.001\ln g} \\ & T_i^{-1.99-0.085\ln T_0-0.027\ln\phi-0.017\ln g} \\ & T_0^{-0.17\ln\phi-1.551+0.155\ln g}\,\phi^{-1.967-0.345\ln g}\,g^{-2.387}. \end{aligned} \tag{23}$$

$$\begin{aligned} N = {} & e^{-0.663}\, I^{1.341-0.066\ln T_i-0.119\ln T_0+0.140\ln\phi+0.845-0.058\ln g} \\ & T_i^{0.230+0.071\ln T_0-0.048\ln\phi+0.016\ln g} \\ & T_0^{-0.197\ln\phi+0.845-0.058\ln g}\,\phi^{0.557-0.003\ln g} g^{0.005}. \end{aligned} \tag{24}$$

To realistically reduce the size of the search space in this problem, it was assumed that the selection of the optimal machining conditions is fixed for the workpiece with diameter $\phi = 68$ mm and depth $g = 6$ mm. Furthermore, the remaining three design variables were subject to the following constraints:

$$\begin{aligned} 500 \le T_i \le 2000 \\ 64 \le I \le 128 \\ 125 \le T_0 \le 250. \end{aligned} \tag{25}$$

The preparation of the training and testing samples in this example was performed as described in Section 4.1. After completion of the learning phase, relative regression errors with 95% confidence were computed.

The range of responses for the four objective functions in the EDM process along with the predicted and verified Pareto-optimal solutions are summarized in Tables 1 and 2, respectively. Note that the first solution appearing in Table 2 was obtained by Osyczka using an ordinary vector optimization technique. Our solutions, on the other hand, are superior to conventional solutions in that they clearly identify optimal design regions with a predicted target value for each of the four objectives.

To test the true nettle of our technique, we decided to verify the predicted optimum solutions by generating an arbitrarily large number of designs within each recommended design surface and compute the means and standard deviations of the resulting responses. For example, consider the first regression solution that appears in Table 2. Here, we predict that the three design parameters of discharge current amplitude (I), pulse duration (T_i), and pulse interval (T_0) can optimally vary within the following intervals:

$$\begin{aligned} & 64 \le I \le 72 \\ & 1732 \le T_i \le 2000 \\ & 212 \le T_0 \le 250, \end{aligned}$$

and that the resulting vector response is (2433, 0.23, 3758, 0.170×10^{-3}). We verified this solution by generating approximately 2000 design points within the predicted optimum design surface and found the means and standard deviations of the responses to be (2490 ± 154, 0.27 ± 0.03, 3777 ± 141, $0.193 \times 10^{-3} \pm 0.014 \times 10^{-3}$). It is evident that designers now have a clear view of what ranges of values can be selected for the design variables. And more im-

Table 1. *Range of responses in the feasible region ([64,128], [500, 2000], [125, 250])*

Response	Minimum	Maximum
Q_v	1994	6797
δ	0.21	13.14
N	2838	7202
R_a	0.000161	0.0012

Table 2. *Predicted versus verified optimal solutions generated for the EDMP example*

Method	Optimum settings for design variables (I, T_i, T_0)	Predicted Pareto-optimal *Verified Pareto-optimal* $(Q_v, \delta, N, R_a \times 10^{-3})$
Osyczka	(64, 2000, 125)	(2458, 0.24, 3727, 0.376)
		(2458, 0.24, 3727, 0.376)
Multivariate regression tree (4 solutions)	([64, 72], [1732, 2000], [212, 250])	(2433, 0.23, 3758, 0.170)
		(2490 ± 154, 0.27 ± 0.03, 3777 ± 141, 0.193 ± 0.014)
	([64, 69], [1732, 2000], [162,187])	(2475, 0.24, 3774, 0.253)
		(2490 ± 95, 0.27 ± 0.02, 3777 ± 91, 0.270 ± 0.017)
	([64, 77], [1732, 2000], [237,250])	(2559, 0.24, 3874, 0.167)
		(2585 ± 218, 0.28 ± 0.03, 3862 ± 195, 0.183 ± 0.009)
	([64, 73], [1732, 2000], [187,212])	(2508, 0.24, 3815, 0.217)
		(2562 ± 159, 0.28 ± 0.03, 3835 ± 146, 0.232 ± 0.014)

portantly, variations within the design surface (various design scenarios) produce process responses, all of which are completely within tolerable margins from their expected target values. The other three solutions in Table 2 were also generated in a similar fashion and further demonstrate the utility of our approach. Figure 4 depicts the design regions identified by these four solutions.

Finally, to compute the accuracy of the generated solutions, we repeated the regression error analysis a total of five times for learning samples of size 200, which resulted in an overall error rate of 0.05 ± 0.0103 with 95% confidence. The four regression solutions that appear in Table 2 were extracted from one of the five generated trees with a regression error of 0.038 ± 0.0087. In general, most of the

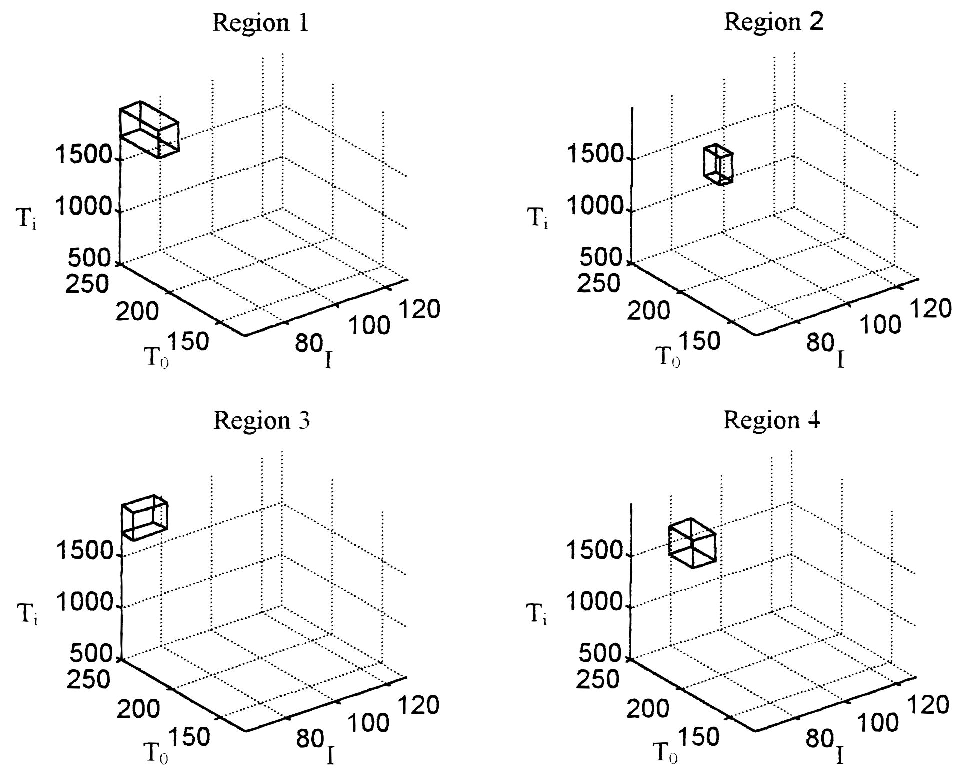

Fig. 4. The four optimal design regions from Table 2.

induced trees produced accurate and optimal designs, most of which were omitted due to space limitation.

5. CONCLUSION

In this paper, we introduced a new methodology within which the problem of multiobjective optimization is transformed into induction of multivariate regression trees. In contrast to a host of conventional techniques, our approach to optimization offers several advantages: 1) an induced regression tree maps hyper-rectangular design regions to minimal-volume ellipsoids within the space of objectives and constraints; 2) because of the minimal-volume criterion, the designed products and processes are least sensitive to unexpected design perturbations; 3) the generalizations obtained by a regression tree allow designers to have a better understanding of complex interrelationships that exist between design variables and product responses; and 4) an *a posteriori* analysis of a generated tree helps locate noninferior regions where a product's performance indices conform to designers' preferences.

To illustrate advantages of our approach, we presented a multiobjective design problem that dealt with optimization of an EDM process and compared our results to those obtained through conventional vector optimization.

ACKNOWLEDGMENTS

The author gratefully acknowledges helpful comments and suggestions of anonymous reviewers.

REFERENCES

Abdel-Malek, H.L., & Hassan, A.S.O. (1991). The ellipsoidal technique for design centering and region approximation. *IEEE Transactions on Computer-Aided Design (10)8*, 1006–1013.

Bozdogan, H. (1990). On the information-based measure of covariance complexity and its application to the evaluation of multivariate linear models. *Communication in Statistics—Theory & Methodology (19)1*, 221–279.

Bradley, S., & White, S.C. (1991). Multiobjective A*. *Journal of ACM (38)4*, 775–814.

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, J.S. (1984). *Classification and regression trees*. The Wadsworth & Brooks/Cole Advanced Books & Software, New York.

Carbonell, J.G., Michalski, R.S., & Mitchel, T.M. (1987). *Machine learning: An artificial intelligence approach*. Tioga Publishing Company, Palo Alto, CA.

Chankong, V., & Haimes, Y.Y. (1983). *Multiobjective decision-making: Theory and methodology*. North Holland, New York.

Everitt, B. (1974). *Cluster analysis*. John Wiley & Sons, Inc., New York.

Fayyad, U.M., & Irani, K.B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning 8(1)*, 87–102.

Giovanni, M. (1983). Response surface methodology and product optimization. *Food Technology 11*, 41–45.

Gnanadesikan, R. (1977). Methods for statistical data analysis of multivariate observations. John Wiley & Sons, Inc., New York.

Hahn, G.J. (1971). How abnormal is normality? *Journal of Quality Technology 3 (1)*, 18–22.

Harris, R.J. (1985). *A primer of multivariate statistics*, pp. 331–333. Academic Press, Inc., New York.

Hwang, C., & Masud, A.S. (1979). Multiple objective decision making: Methods and applications. In *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, New York.

Lokanathan, A.N., Brockman, J.B., & Renaud, J.E. (1995). A multidisciplinary optimization approach to integrated circuit design. *University of Notre Dame, Computer Science Technical Report Series 95-5*, pp. 1–21.

Lokanathan, A.N., Brockman, J.B., & Renaud, J.E. (1996). A Methodology for Concurrent Fabrication Process/Cell Library Optimization. *Proc. 33rd Ann. Conf. Design Automation Conf.*, 825–830.

Mehrotra, K.G., Mohan, C.K., & Ranka, S. (1991). Bounds on the number of samples needed for neural learning. *IEEE Transactions on Neural Networks 2(6)*, 548–558.

Murthy, S. (1995). On growing better decision trees from data. PhD Dissertation. Johns Hopkins University Press, Baltimore, MD.

Myers, R.H., Khuri, A.I., & Carter, W.H. (1989). Response surface methodology: 1966–1988. *Technometrics 31(2)*, 137–157.

Osyczka, A. (1984). *Multicriterion optimization in engineering*. Ellis Harwood Limited. Chichester, W. Sussex, UK.

Phadke, M. (1989). *Quality engineering using robust design*. Prentice Hall, Englewood Cliffs, NJ.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning 1*, 81–108.

Rendell, L. (1990). Induction as optimization. *IEEE Transactions on Systems, Man, and Cybernetics 20(2)*, 324–338.

Shavlik, J.W., & Dietterich, T.G. (1990). *Readings in machine learning*. Morgan Kaufmann Publishers, Inc., Los Altos, California.

Shien, D.S., & Joseph, B. (1992). Exploratory data analysis using inductive partitioning and regression trees. *Industrial Chemical Engineering Research 31*, 1989–1998.

Tatsuka, M.M. (1971). *Multivariate analysis: Techniques for educational and psychological research*. John Wiley & Sons, New York.

Weiss, S.M., & Kulikowski, C.A. (1991). *Computer systems that learn*. Morgan Kaufmann Publishers, Inc., Los Altos, California.

Zwick, R. (1985). Nonparametric one-way multivariate analysis of variance: A computational approach based on Pilla–Bartlett Trace. *Psychological Bulletin 97(1)*, 148–152.

Babak Forouraghi received his Ph.D. in computer science in 1995 from Iowa State University where he was a member of the Artificial Intelligence Group at the Center for Nondestructive Evaluation. He is currently an assistant professor of computer science at Saint Joseph's University. His research interests include database mining, fuzzy logic, robust design, and use of decision trees, regression trees, and genetic algorithms in off-line quality-control applications.