Optimal Engineering System Design Guided by Data-Mining Methods

Pansoo KIM

School of Business Administration
Kyungpook National University
Daegu, South Korea, 702-701
(*pskim@knu.ac.kr*)

Yu DING

Department of Industrial Engineering
Texas A&M University, 3131 TAMU
College Station, TX 77843-3131
(*yuding@iemail.tamu.edu*)

An optimal engineering design problem is challenging because nonlinear objective functions usually need to be evaluated in a high-dimensional design space. This article presents a data-mining–aided optimal design method, that is able to find a competitive design solution with a relatively low computational cost. The method consists of four components: (1) a uniform-coverage selection method, that chooses design representatives from among a large number of original design alternatives for a nonrectangular design space; (2) feature functions, of which evaluation is computationally economical as the surrogate for the design objective function; (3) a clustering method, that generates a design library based on the evaluation of feature functions instead of an objective function; and (4) a classification method to create the design selection rules, eventually leading us to a competitive design. Those components are implemented to facilitate the optimal fixture layout design in a multistation panel assembly process. The benefit of the data-mining–aided optimal design is clearly demonstrated by comparison with both local optimization methods (e.g., simplex search) and random search-based optimizations (e.g., simulated annealing).

KEY WORDS: Classification and regression tree; Fixture layout optimization; K-means clustering; Kriging model; Multistation assembly processes; Uniform design.

1. INTRODUCTION

An optimal design problem is to select the best of alternative design variables from a candidate design space subject to certain constraints. Generally, an optimal design problem needs to evaluate nonlinear objective functions in a high-dimensional design space. Nonlinear programming methods, such as sequential quadratic programming (Hillier and Lieberman 2001) and simplex search (Nelder and Mead 1965), have been used to find the optimal solution, and they usually converge to a solution in a relatively short time. But the quality of the final solution depends highly on the selection of an initial design. These methods are known as "local" optimization methods, because the solutions are easily entrapped in a local optimum. To escape local optima, one would prefer to use a random search-based method, such as the genetic algorithm (GA) (Gen and Cheng 2000) or the simulated annealing algorithm (SA) (Bertsimas and Tsitsiklis 1993). Empirical evidence shows that GA and SA are indeed quite effective in escaping local optima but at the expense of considerably slower convergence, and thus are impractical when the computation cost of evaluating an objective function is high (Schwabacher, Ellman, and Hirsh 2001).

As an example of optimal engineering design, we consider the assembly process of the side frame of a sport utility vehicle (SUV) in Figure 1. The final product, the *inner-panel complete*, comprises four components (A-pillar, B-pillar, rail roof side panel, and rear quarter panel) assembled on three stations (stations I, II, and III). Then the final assembly is inspected at station IV [M_1–M_{10} in Fig. 1(d) are key dimensional features]. The dimensional quality measured at those key features is determined mainly by the variation input from fixture locators P_1–P_8. The design objective is to find the optimal fixture layout of a multistation assembly process so that the product-dimensional variability (measured at M_1–M_{10}) is insensitive to the fixture variation input.

There are eight fixture locators (P_1–P_8) in the aforementioned assembly process. Each part or subassembly is positioned by a pair of locators. For the sake of simplicity, we are concerned only with a two-dimensional (2-D) assembly in the X–Z plane. As illustrated in Figure 1(e), two different types of locator are used in the locating pair on each panel in the fixturing mechanism under consideration. The pin-hole locator, usually called a four-way locator, will restrain two degrees of freedom (df) of the part motion, and the pin-slot locator, usually called a two-way locator, will restrain the remaining 1 df of the 2-D part. Thus this pair of locators can provide a complete constraint on 3 dfs that any 2-D part has.

In a 2-D assembly process, the position of a locator is determined by its X and Z coordinates. Thus the design space has 16 parameters for 8 locators and is continuous, meaning that there are infinite numbers of design alternatives. We can generate a finite candidate design space via discretization, say, using a resolution of 10 mm (the size of a locator's diameter) on each panel. This resolution level will result in the number of candidate locations on each panel as $n_1 = 308$, $n_2 = 905$, $n_3 = 396$, and $n_4 = 6,204$, where n_j denotes the number of candidate locations on panel j. Because of the differences of the two locators used on each panel, the layout of $P_1 = (X_1, Z_1)$ and $P_2 = (X_2, Z_2)$ may generate different responses on product-dimensional variability than the layout of $P_1 = (X_2, Z_2)$ and $P_2 = (X_1, Z_1)$ does. Thus the total number of design alternatives is $2C_2^{308} \times 2C_2^{905} \times 2C_2^{396} \times 2C_2^{6,204} \approx 4.65 \times 10^{23}$, where C_a^b is the combinational operator—namely, it is the number of ways in which a objects can be selected from a set of size b. Apparently, the number of design alternatives is overwhelmingly large, and numerous local optima are embedded in the

TECHNOMETRICS, AUGUST 2005, VOL. 47, NO. 3
DOI 10.1198/004017005000000157

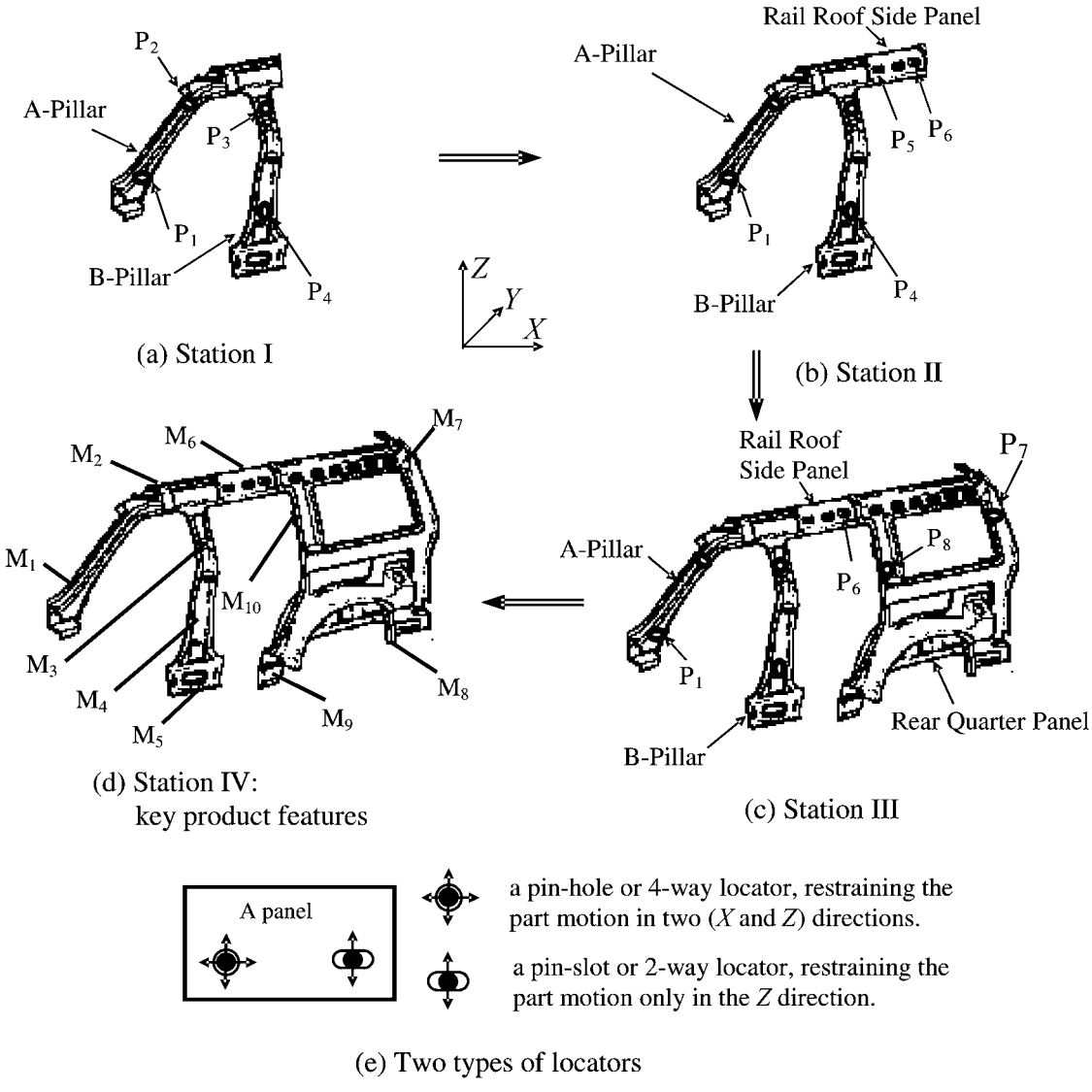

Figure 1. Assembly Process of an SUV Side Frame.

16-dimension design space. Any local optimization method will hardly be effective, and a GA/SA random search could be inefficient.

Recently, noteworthy efforts have been made in using data-mining methods to aid the process of an optimal engineering design (Schwabacher et al. 2001; Igusa, Liu, Schafer, and Naiman 2003). The basic idea is to use a data-mining method—various classification methods are the major ones used in such applications—to extract good initial designs from a large volume of design alternatives. In other words, if the design alternatives are treated as a dataset, then a data-mining method may be able to discover valuable structures within the dataset and generalize design selection rules leading us to a much smaller good design subset, which is more likely to yield a better design solution even if a local optimization method is applied.

The idea is illustrated in Figure 2. A data-mining method generalizes the design selection rules based on the training data in a design library, which in turn is created either from a collection of historical design results or from random sampling among design alternatives. The resulting selection rules are often expressed as a classification tree or, equivalently, a set of "if–then" rules. Then the large number of design alternatives will pass through the selection rules, and certain local optimization methods will be applied to the selected good designs to find the final optimal design. Schwabacher et al. (2001), for instance, applied this idea in the prototype selection of structures of a racing yacht and a supersonic aircraft, where their design li-

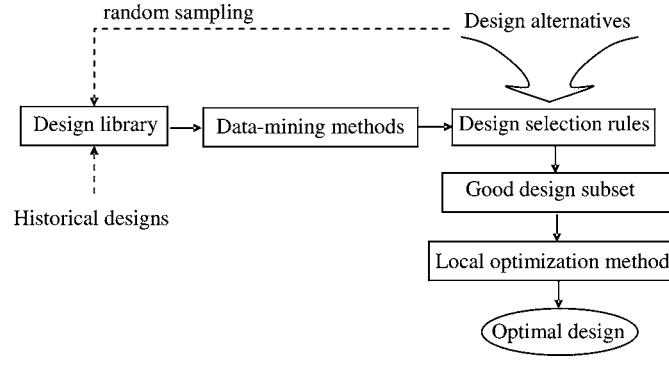

Figure 2. Design Optimization Using Data-Mining Methods.

brary is created from historical designs and a C4.5 decision tree (Quinlan 1993) is used to generate the design selection rules.

Although the general idea as described in Figure 2 could aid in discovering valuable design selection guidelines, there is a major obstacle to applying this idea to engineering design problems, especially those with a computationally expensive objective function. The obstacle is that for a new design without sufficient historical data, generation of design selection rules needs to evaluate the objective functions of all designs in a design library. For the design library to be representative of a large volume of design alternatives, one will have to include a sufficient number of designs in the library—potentially too many to be computationally affordable for generating the design selection rules.

In the design of a civil structure, Igusa et al. (2003) proposed a more sophisticated idea that circumvents frequent evaluation of an objective function. They recommend using a much simpler feature function together with a clustering method to reduce the number of designs whose objective function needs to be evaluated for the generation of a classification tree.

Following the general idea proposed by Igusa et al. (2003), we develop a data-mining–aided optimization method for the aforementioned multistation fixture layout design. The method includes the following components: (1) a uniform-coverage selection method that chooses design representatives from among a large amount of original design alternatives for a nonrectangular design space; (2) feature functions; of which evaluation is computationally economical as the surrogate for the design objective function; (3) a clustering method that generates a design library based on the evaluation of feature functions instead of an objective function; and (4) a classification method to create the design selection rules, eventually leading us to a competitive design. We demonstrate the design effectiveness and efficiency of the proposed method by comparison with the simulated annealing algorithm and a local optimization method.

Among the aforementioned four components, identifying good feature functions as a surrogate for the true objective function plays a critical role. Existence of such a set of feature functions is actually the reason why we can eventually reduce computation. However, identification of feature functions is still ad hoc and a matter of engineering knowledge. For this reason, we caution the reader that the success of the proposed methodology depends greatly on the presence and reliability of such knowledge.

The article is organized as follows. Section 2 presents the data-mining–aided design method and develops each component in the context of the multistation fixture layout design. Section 3 implements the resulting method and compares the algorithm performance with other optimization methods. Finally, Section 4 concludes the article.

2. DATA–MINING–AIDED OPTIMAL DESIGN METHOD

2.1 Design Objective and Overview of the Method

The goal of fixture layout design, as already mentioned in Section 1, is to find an optimal layout so that assembly-dimensional variability is insensitive to variation of inputs from fixture locators. Research efforts have been made to establish a linear variation propagation model that links the product-dimensional deviation (measured at M_1–M_{10}) to fixture locator deviations at P_1–P_8 on three assembly stations (Jin and Shi 1999; Ding, Ceglarek, and Shi 2000; Camelio, Hu and Ceglarek 2003). Based on the variation model, a sensitivity index S was developed by Kim and Ding (2004) as a nonlinear function of the coordinates of fixture locators, represented by the 16×1 parameter vector $\boldsymbol{\theta} \equiv [X_1 \quad Z_1 \quad \cdots \quad X_8 \quad Z_8]^T$, where X_i and Z_i are the pair of coordinates of locator P_i. Using this notation, the fixture layout design attempts to find a set of $\boldsymbol{\theta}$ that minimizes the sensitivity S while satisfying the geometric constraint $G(\cdot)$, that is,

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & S(\boldsymbol{\theta}), \\ \text{subject to} \quad & G(\boldsymbol{\theta}) \geq 0. \end{aligned} \tag{1}$$

Equation (1) actually captures a general formulation of a nonlinear optimization problem; $S(\cdot)$ is the objective function, $G(\cdot)$ is the constraint function, and $\boldsymbol{\theta}$ is the vector of design parameters. In the foregoing formulation, without loss of generality, we present a minimization problem. A maximization problem can be solved in the same fashion. The objective function $S(\cdot)$ in an engineering optimal design is generally complicated. For example, if we consider the flexibility of parts during assembly operations, then $S(\cdot)$ will have to be evaluated using computationally expensive finite-element analysis (FEA) codes (Camelio et al. 2003). For this reason, the efficiency of an optimal design algorithm can be loosely determined by how often $S(\cdot)$ is evaluated—we denote by T the computer time necessary to calculate $S(\cdot)$ once.

There is virtually no efficient method that allows us to directly optimize over the huge volume of original design alternatives, such as the possible combinations of locators, as many as 4.65×10^{23}, in the multistation fixture layout design. The proposed method starts with extracting *design representatives* from original design alternatives. However, it is often the case that the design representatives, although remarkably fewer than the original design alternatives, are still too many to be used as the design library. In this article we use a clustering method with a set of computationally simple feature functions to facilitate the creation of a design library. This procedure will allow us to eventually have an affordable number of designs as a training dataset in a design library. The overall framework is illustrated in Figure 3, as a modification to Figure 2. In the following sections we present the considerations and procedures for realizing each component of the data-mining–aided optimal design.

2.2 Candidate Design Space

Before getting into the details of the proposed design method, we first describe the design space for candidate locators (called the "candidate design space"). The candidate design space imposed by $G(\cdot)$ in (1) is different from the natural boundary of each panel. The boundary of the candidate design space should be at least 35 mm away from the edge of a panel due to an engineering safety requirement, because a locating hole that is too close to the edge may not be able to ensure the load exerted during fixturing. In addition to this, we know that a part-positioning

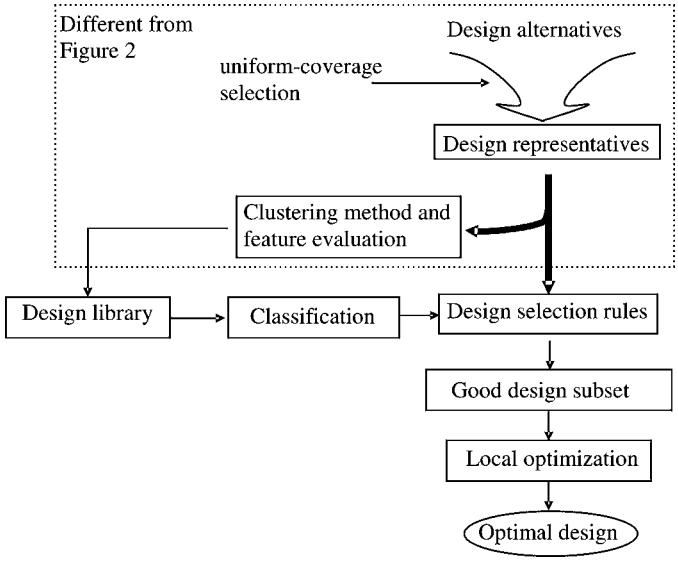

Figure 3. Modified Data-Mining–Aided Design Optimization Procedure.

deviation is more sensitive to locating deviations when both locators are close to each other than when they are farther apart. This rule suggests that two locators on the same panel should be located sufficiently far away from each other, which rules out the neighborhood around a panel's geometric center (GC) from consideration for the candidate design space.

The determination of this neighborhood is illustrated in Figure 4(a). Distances from the GC to the vertex points of a panel are calculated, and their median value is chosen to represent the size of the panel, denoted by d_0. A hypothetical circle is drawn on the panel with the GC as its center and $d_0/2$ as its radius. The area inside this hypothetical circle is considered the neighborhood of a panel's geometric center. Using the median of all GC-to-vertex distances in determining d_0, rather than their mean value, makes the resulting d_0 less sensitive to a very large or a very small GC-to-vertex distance on panels with an irregular shape. (Recall that a median is a more robust statistic than a mean value.) The value of $d_0/2$ is an empirical choice. Our experience indicates that the choice is actually quite conservative; that is, after removing candidate locations using $d_0/2$ as the neighborhood radius, we did not see much difference in terms of the best found sensitivity value of fixture layouts (Kim and Ding 2004). One can certainly decrease the neighborhood radius to be safer. The resulting candidate design space imposed by $G(\cdot)$ is shown as the shaded areas in Figure 4(b), to which all of the latter optimal design methods will be applied and their performances compared.

If the candidate design space of each panel is discretized using the same 10-mm resolution, then the numbers of candidate fixture locations are $n_1 = 239, n_2 = 707, n_3 = 200$, and $n_4 = 3{,}496$. After eliminating the locator pairs of which distances are smaller than $d_0/2$ on each panel, we still have as many as 1.09×10^{21} possible combinations of locator layouts. This number is still too large to be computationally affordable for design optimization.

2.3 Uniform Coverage Selection of Design Representatives

The first component of the proposed method is to select design representatives from the original design alternatives. Unless one has profound knowledge of which part in the candidate design space (after the neighborhood of a geometric center has been ruled out) is preferred, a safer way of selecting good representatives of the original design set is to select them from a design space as evenly as possible. Igusa et al. (2003) suggested randomly selecting design representatives, based on a uniform distribution, from the set of design alternatives. The problem with random selection is that probabilistic uniformity does not guarantee an even geometric coverage in a design space. When the design space is of a high dimension and the sample size is relatively small (e.g., 2,000 chosen from 8.5×10^8 alternatives in Igusa's case), the selected sample could cluster in a small area and fail to cover large portions of the design space (Fang and Wang 1994).

A space-filling design, widely used in computer experiments (Santner, Williams, and Notz 2003), aims to spread design points evenly throughout a design region and appears to fit well into our purpose of design-representative selection.

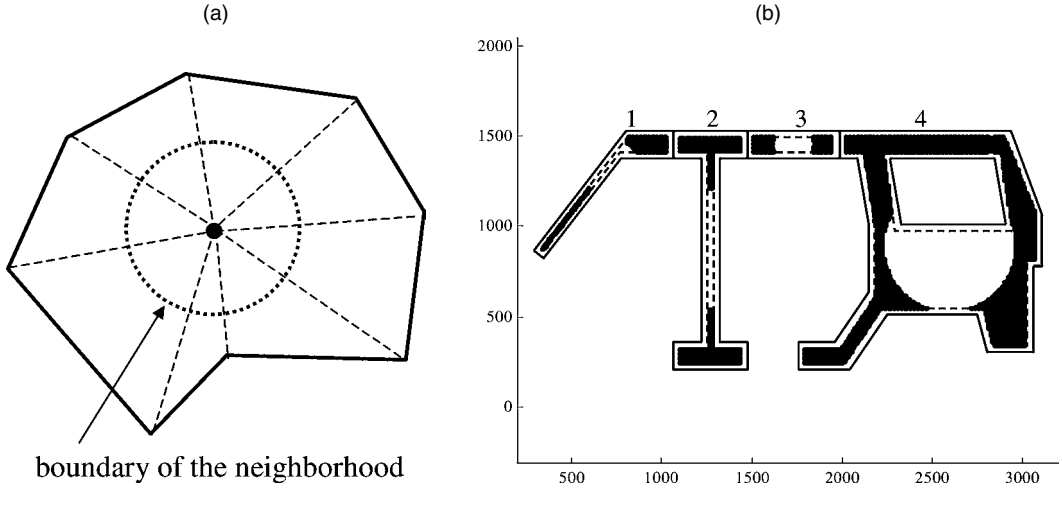

Figure 4. Neighborhood of a GC (a) and Candidate Design Space on SUV Side Frames (b).

A space-filling design is usually devised by using latin hypercube sampling (LHS) (McKay, Bechman, and Conover 1979), a stratified sampling method, or using a uniformity criterion from the number-theoretic method (NTM) (Fang, Lin, Winkle, and Zhang 2000).

These methods can be easily implemented over a hyperrectangular design space in experimental designs. In engineering system designs, accompanied by complicated geometric and physical constraints, the design space is often nonrectangular or even highly irregular, such as the candidate design space shown in Figure 4(b). Another constraint also comes into play in this fixture layout design; that is, once a locator's position is chosen on a panel, the second locator on the same panel should not be located near the first one, following the same physical intuition related to positioning variability explained previously. This is different from the factor-level selection in experimental designs, where there is usually no clear prior knowledge to indicate the dependency among factors.

Given the complexity in the design constraints, we have not seen a generic method for translating an LHS- or NTM-based space-filling design to an engineering system design problem. We here devise a heuristic procedure for the fixture layout design, attempting to provide a uniform-coverage selection of design representatives from the original fixture layouts:

Step 1. Uniformly discretize the candidate design space on each panel plane using the same resolution. (In our implementation, the resolution is 10 mm between two adjacent locations.) Associate a probability p to each candidate location, with p initially set to be equal for all locations.

Step 2. On each panel, choose the first locator sequentially to be at those locations from the discretization process. Once the first locator is selected, generate a subset of locations containing those with a distance from the first locator greater than half of the panel size ($d_0/2$). The second locator is selected according to the probability associated with all of the locations in that subset. Once a location is chosen, update the probability for this location, $p_{\text{new}} = \gamma \cdot p_{\text{old}}$, $0 < \gamma < 1$; namely, reduce the probability of selecting this location by a factor γ. Denote by $\Omega_j^{(0)}$ the resulting candidate locator set for panel j. Then n_j equals the number of locator pairs included in $\Omega_j^{(0)}$.

Step 3. For $i = 1, \ldots, \max_j(n_j)$,

(a) randomly select one locator pair from each of $\Omega_j^{(i-1)}$ for $j = 1, 2, 3, 4$ *without replacement*;

(b) combine these four locator pairs as one design representative; and

(c) whenever a $\Omega_j^{(i-1)}$ becomes empty, reset $\Omega_j^{(i-1)} = \Omega_j^{(0)}$; otherwise, set $\Omega_j^{(i)} = \Omega_j^{(i-1)}$, $j = 1, 2, 3, 4$.

End of the loop.

Figure 1(e) shows that two different types of locators are used as a locator pair on each panel. In the foregoing selection procedure, the four-way locator restraining 2 df is considered more important, so it is selected as the first locator in Step 2; its uniformity is a result of the uniform discretization. The two-way locator is treated as the second locator and is chosen to be at least $d_0/2$ away from the first locator because of the aforementioned constraint on the between-locator distance. The threshold of $d_0/2$ is again chosen empirically, following the same spirit as in Section 2.2. The reason that we associate a probability with each location and subsequently reduce the probability for a selected location is to ensure that the second locator is more likely to be evenly spread out. Had we used a simple random selection for choosing the second locator, then there would be a greater chance that values of the second locator position would form clumps or clusters of values. With the discount factor γ set to be .1 in our implementation, once a location is selected, the probability of choosing it again will be 10% of the original probability, and choosing it the third time will be very unlikely (1% of the original probability).

After Step 2, the set $\Omega_4^{(0)}$ has the largest number of locator pairs, $n_4 = 3{,}496$. Step 3 performs a stratified sampling to generate locator combinations. The stratified sampling will go over $\Omega_4^{(0)}$ once but will have to go over $\Omega_j^{(0)}$ for panel $j = 1, 2, 3$ multiple times. Step 3 can be thought of as being equivalent to what will be achieved by the following procedure: First, augment $\Omega_j^{(0)}$ for $j = 1, 2, 3$ to be the same size as $\Omega_4^{(0)}$, then perform a LHS on this four-dimensional rectangular region. Hence this step can be considered a generalization of LHS to nonregular regions.

Eventually, a total of 3,496 combinations of locators is generated as design representatives. We denote this number by N_r.

2.4 Feature Definition and Feature Function Selection

To avoid direct and frequent evaluations of objective function $S(\cdot)$, we use a set of feature functions to characterize the system performance. A feature function maps an engineering system to a feature that is tied to the design objective. For example, the distance between two locators in the fixture design can be considered a feature. Generally, any physical quantity that is potentially tied to the design objective can be used as a feature. The set of feature functions is actually a surrogate for the design objective function. Features are often selected based on prior experience, engineering knowledge, or physical intuition. The advantage of such a feature definition/selection is that vague experience, knowledge, or understanding of a complicated engineering system can be more systemically integrated into the optimal design process.

What follows feature selection is a clustering method acting on the set of the chosen feature functions. If the feature function does not form well-separated clusters, then the subsequent clustering step will not be effective when it attempts to form a design library with smaller number of designs. This problem may very well be avoided if the feature functions are chosen to be the actual surrogate for the objective function. If the objective function forms a lot of local optima on the response surface, then the feature functions also will be likely to form clusters, corresponding to those local optimum areas. Therefore, although the selection of a feature in the proposed method is rather flexible, we actually need to do so with care.

We have the following generic considerations for an effective selection of features and feature functions. First, when choosing the feature functions, we need to make sure that they are directly related to the objective function instead of making them the replacements of design variables in $\boldsymbol{\theta}$. In particular, we need to avoid choosing feature functions that are just subsets of $\boldsymbol{\theta}$ or are linearly related to $\boldsymbol{\theta}$. Second, because features are used to replace the direct evaluation of an objective function, a feature function should be computationally simple; otherwise, it will not serve our purpose. Third, because a feature is usually not connected to the design objective with mathematical explicitness, too few feature functions may generate a serious bias in the latter selection of design representatives. On the other hand, too many feature functions will increase the computation burden. A trade-off will depend on specific applications, where between 5 and 15 feature functions may be selected. Finally, it is desirable to select scalable features; that is, a feature definition will remain the same when the size of a system has increased. For the example of the multistation fixture design, a scalable feature means that it can be used to characterize the system performance whether the multistation system has 3 stations or 10 stations.

Keeping in mind the foregoing guidelines, we choose a set of feature functions for the fixture layout design as follows. We know that the distance between locators is an important factor related to the variation sensitivity of a fixture layout. We select the between-locator distance on a panel as one feature relevant to our design objective. We select the following five functions to characterize the feature of between-locator distance (the five feature functions actually approximate the distribution of the set of the between-locator distances):

$F_1(\boldsymbol{\theta})$, the largest value of the same-panel between-locator distances

$F_2(\boldsymbol{\theta})$, the second largest value of the same-panel between-locator distances

$F_3(\boldsymbol{\theta})$, the mean of the same-panel between-locator distances

$F_4(\boldsymbol{\theta})$, the second smallest value of the same-panel between-locator distances

$F_5(\boldsymbol{\theta})$, the smallest value of the same-panel between-locator distances.

For a larger-scale assembly system with more parts and stations, these feature functions can still be used; namely, they are scalable. The approximation of the distribution could be improved by augmenting the number of feature functions so that they will give more refined percentile values of the set of between-locator distances.

If we are concerned only with a single part that is positioned by a pair of locators at a single station, then the between-locator distance may be the only factor that matters. However, complexity results from the fact that locating holes on a panel are reused, but usually in a different layout. For the multistation assembly process in Figure 1, the A-pillar and B-pillar are positioned on station I by $\{P_1, P_2\}$ and $\{P_3, P_4\}$. After the assembly operation is finished, the subassembly becomes one single piece, which is transferred to station II and positioned by $\{P_1, P_4\}$. This assembly transition across stations and the reuse of fixture-locating holes complicate the sensitivity analysis for a multistation system. Kim and Ding (2004) showed that a larger between-locator distance on one station may not necessarily produce a lower sensitivity for the whole process. In order to capture the across-station transition effect, we select a second feature, which is the ratio of between-locator distances on two adjacent stations. Denote by $L_1, L_2, \ldots, L_m$ the between-locator distance for m locator pairs on a station. After those parts are assembled, they are transferred to the next station and positioned by a locator pair with a between-locator distance $L^{(m)}$. The ratio of distance change r is then defined for this transition as

$$r \equiv \frac{L^{(m)}}{(\sum_{i=1}^{m} L_i)/m}. \tag{2}$$

Here we include three more feature functions related to the feature of distance change ratio as:

$F_6(\boldsymbol{\theta})$, the largest value of distance change ratios

$F_7(\boldsymbol{\theta})$, the mean value of distance change ratios

$F_8(\boldsymbol{\theta})$, the smallest value of distance change ratios.

Similarly, the three feature functions approximate the distribution of the set of r. We do not include five functions as we did for the between-locator distances, because four stations in this example produce only three distance change ratios.

We have defined eight scalable feature functions for two physically intuitive features relevant to the variation sensitivity of a multistation assembly process. Note that the calculation of the foregoing feature functions is very economical even for a large-scale system.

2.5 Clustering Method

Clustering aims to segment a heterogeneous population into a number of more homogeneous subgroups (Hastie, Tibshirani, and Friedman 2001). When using feature functions as the surrogate for a design objective to benchmark the dissimilarity criteria in a clustering procedure, design representatives in a resulting cluster will have a more similar distribution profile for the two physical features, namely the between-locator distance and the across-station distance change ratio. Empirical evidence (Igusa et al. 2003) shows that resulting clusters are associated with a local response surface and that its center will likely be around a local optimum. For this reason, a design library for classification can then be created by selecting a few designs from each cluster around the cluster center, which results in fewer designs. The generation of a design library is illustrated in Figure 5.

For the ith fixture layout represented by $\boldsymbol{\theta}_i$, $\mathbf{F}_i \equiv [F_1(\boldsymbol{\theta}_i) \cdots F_8(\boldsymbol{\theta}_i)]^T$ is the vector of its feature functions and $c(i)$ denotes the cluster to which it belongs. In our solution procedure we use a standard *K-means clustering* method (Hastie et al. 2001). Namely, for K clusters we find the mean value of cluster k, $\mathbf{m}_k$, as the cluster center and the association of a fixture layout to cluster k [represented by $c(i) = k$] so that

$$\min_{c, \{\mathbf{m}_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{c(i)=k} \|\mathbf{F}_i - \mathbf{m}_k\|^2, \tag{3}$$

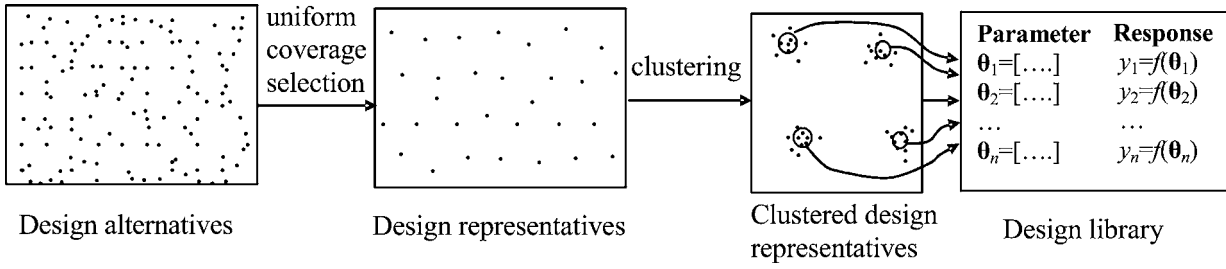

Figure 5. Generation of a Design Library.

where N_k is the number of elements in cluster k and $\|\cdot\|$ is a vector 2-norm. The $K-$means method minimizes the dissimilarity measure, defined as the Euclidean distances of the elements within the same cluster. With different values of K, the minimization in (3) will yield different clustering results, that is, different cluster centers and cluster associations. We defer the discussion of how to choose K to Section 2.7.

Once the design representatives are clustered (i.e., each fixture layout is labeled with a cluster identification), we choose a few designs around the cluster center to form a design library, as illustrated in Figure 5. We call the selected designs from each cluster center *seed designs* and let J_k denote the number of seed designs chosen from cluster k. For the sake of simplicity, we use the same seed number for all clusters, that is, $J_k = J$ for all k's. Then the design library contains KJ data pairs $\{\mathbf{F}_i, S_i\}$ for $i = 1, 2, \ldots, KJ$, where S_i is the sensitivity value of the ith fixture layout.

Finally, the clustering step is briefly summarized as follows. We are given N_r design representatives from the uniform-coverage selection; then:

Step 1. For each design, determine $\mathbf{F} = (F_1, \ldots, F_8)$.

Step 2. Cluster the $\mathbf{F}$ values into K clusters using some clustering method (e.g., K-means clustering). The clustering method should be one where the measure of similarity is consistent with the assumption that when two values of $\mathbf{F}$ are similar, the corresponding values of the objective function S are similar. Usually, this will mean Euclidean distance is the measure of similarity.

Step 3. For each of the K clusters, choose the J designs that are closest to the center of the cluster. This yields KJ designs, and one typically wants $KJ \ll N_r$.

2.6 Classification Method

We perform classification on the dataset in the design library to generate the design selection rules. This step is similar to what has been implemented before (Schwabacher et al. 2001, also refer to Fig. 2). Local optimization methods can be used to evaluate a few designs chosen by the selection rules and yield the final optimal design. Many times, as we discuss in Section 3, a local optimization method may no longer be necessary; that is, a direct comparison among all the selected designs could have given us a satisfactory result.

Schwabacher et al. (2001) and other authors fulfilled the classification step using a tree-based method, including a C4.5 decision tree (Quinlan 1993) and a classification and regression tree (CART). To make the proposed method well connected to the relevant work in literature, we first discuss how to use a tree-based method for the classification step. According to Hastie et al. (2001, p. 273), the later version of Quinlan's decision tree is very similar to a CART; here we focus our discussion on CART, because it is widely accessible through a commercial software program; such as MATLAB.

In our problem we choose to apply a CART model to $\{\mathbf{F}_i, S_i\}$ in the design library. The paths in a CART can be expressed as a set of "*if–then*" rules in terms of the feature functions. One resulting classification tree is shown in Figure 6. A decision condition such as $F_6 < 1.15$ is indicated at each node. One takes

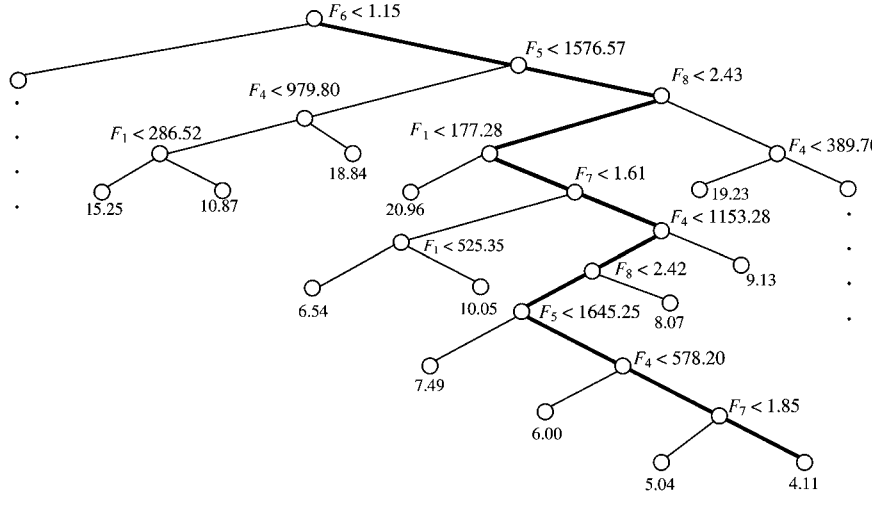

Figure 6. Part of the Classification Tree for the Fixture Layout Design.

the left-side path if the answer to this condition is "yes," and the right-side path if the answer is "no." An end node in the tree represents a set of designs associated with a narrow range of sensitivity values; the value indicated in Figure 6 beside an end node is the average sensitivity value of the corresponding design set. If a certain combination of feature function values leads us to a set of designs whose expected sensitivity value is the lowest among all end nodes, then the corresponding path (one of which is highlighted in Fig. 6) constitutes a design selection rule that we are looking for. The resulting selection rule will be applied to the whole set of N_r design representatives. The designs finally selected constitute a so-called "good design" subset, as indicated in Figures 2 and 3. Note that because of the random selection of the second locator on a panel, the resulting tree is not exactly the same each time that we start the design process over. But our results show that this difference does not cause much difference in the final optimal design.

One critical question is how to select the final tree structure. What we need to achieve here is to find a tree that not only has reasonably good predictive power, but also can generate a substantially small good design subset when it is applied to N_r design representatives. These two objectives are somewhat conflicting. A simple tree structure will usually have a better predictive power but will cause the good design subset undesirably large. In contrast, a full-structure tree without pruning will lead to a smaller good design subset but may lack predictive capability for a new dataset. During our research, we found that applying an existing criterion, such as the one-standard-error rule (Hastie et al. 2001, p. 57) or the cost-complexity index (Hastie et al. 2001, p. 270) in selecting a tree structure usually will lead to a tree that is too simple for our application. This is not surprising, because both criteria attempt to find the simplest tree with a small prediction error; the size of good design subset does not come into consideration. Here we propose a revised cost-complexity index, C, more suitable for our problem,

$$C \equiv \sum_{i=1}^{KJ} L(S_i, \hat{S}_i) + \frac{N_f}{\alpha N_r - KJ}, \tag{4}$$

where $L(S_i, \hat{S}_i)$ is the squared-error loss function, $\hat{S}_i$ is the predicted sensitivity value using the resulting tree, N_f is the number of designs in the good design subset, and $0 < \alpha < 1$ is a prechosen constant. In (4), the cost part (i.e., the loss function) is the same as the one used in the cost-complexity index by Hastie et al. (2001); it characterizes a tree's predictive power and will be evaluated based on a 10-fold cross-validation. The complexity part is replaced by a ratio tied to the size of the good design subset. The αN_r represents the level of how many designs that one would like to evaluate. After subtracting KJ, the number of designs used to generate the design library, the denominator $\alpha N_r - KJ$ provides a reference level for the size of the good design subset. We recommend choosing $\alpha = .1$ or smaller, so that the computation for evaluating those designs will remain affordable. In our problem, the choice of $\alpha = .1$ translates to 350 overall designs, and the final tree structure is selected by applying the one-standard-error rule on this new index C.

An obvious motive for us to use a CART model in this classification step is its common applications in the previous research of data-mining–aided design. It is not merely a coincidence that a CART model has been commonly chosen; it indeed has several advantages over competitive methods. Hastie et al. (2001, p. 313) compared the tree-based method with other methods, including the support vector machine (SVM) and the neural net. The tree-based method is better than other methods in terms of its capability of handling data of mixed types, dealing with irrelevant inputs, being robust to outliers, and also being computationally scalable. These are all desired properties when we are trying to establish a connection between the feature function and the sensitivity function. For example, its better capability of handling irrelevant inputs is important, because feature functions are usually selected based on rough engineering knowledge and we might have selected a few irrelevant ones. Computational scalability is also important, because we may need to select more feature functions to avoid bias when dealing with a large system. In engineering applications, the tree-based method is also popular because it can be easily implemented and the resulting decision rule has an intuitive interpretation. For this reason, Hastie et al. (2001) considered the tree-based method to be one of the best candidates for an off-the-shelf data-mining method.

One disadvantage of a tree-based method is its relatively poor predictive power. In our optimal design procedure, however, we do not use the resulting tree to directly predict the optimal solution. Instead, we end up with a subset of designs with a good likelihood of containing the best design of the N_r design representatives. Evaluations of those designs will make the predictive power of a tree less critical to our finding of an optimal solution. But the poor predictive power of a tree does take a toll on the total number of designs that are eventually evaluated. In addition to the KJ designs in the design library, we must evaluate extra N_f designs in the good design subset.

Had we used a method with a better predictive power, we may have been able to find a nearly optimal design without evaluating a good design subset. During our research, we explored two methods with better predictive power: the boosting tree and the kriging model. The boosting tree is a natural extension of the tree-based method with a usual regression tree as the base learner. We adopted a gradient tree boosting for multiple additive regression tree (MART), a detailed procedure for which has been given by Hastie et al. (2001, p. 322). The kriging model is a popular predictive model that has been broadly used in computer experiment research (Santner et al. 2003). Through our investigations, we found that the MART and kriging models are able to find a reasonably good design from design representatives without evaluating the good design subset. (The numerical comparisons are given in Table 3 in Sec. 3.) For the fixture layout design at hand, these two predictive methods perform similarly, and the best designs found are somehow worse than the one found by using the procedure of a CART model plus subsequent subset evaluation. Nonetheless, for the situations when any saving in objective function evaluation would be appreciated, using these high-predictive-power methods may be preferred over a simple CART model. Because of the similar performance of the MART and kriging models, we are in favor of MART in our application because it inherits the advantages of tree-based methods and it is easier for practitioners to implement.

One may wonder whether we could use the aforementioned methods to fit a model for $\{\boldsymbol{\theta}_i, S_i\}$ and then find the values of $\boldsymbol{\theta}$ that (nearly) minimize the sensitivity. We generally discourage using a CART model for connecting $\boldsymbol{\theta}$ to S. The response surface of $S(\boldsymbol{\theta})$ is presumably complicated and nonlinear, and it is usually unlikely that the surface is well approximated by a binary partition procedure, which tree-based methods use. Using feature functions will cause less problems. If the feature functions, as the surrogate for the sensitivity, can be sensed by human experts, it is more likely to be well modeled by a tree structure, because a tree mimics the way a human expert (e.g., an engineer or a medical doctor) thinks. Predictive methods such as a kriging model, an SVM, or a neural net can fit a much more complicated response surface and may be more suitable in fitting a model for $\{\boldsymbol{\theta}_i, S_i\}$. However, we are not in favor of using a kriging model or the like to fit a model for $\{\boldsymbol{\theta}_i, S_i\}$, either, because of two obvious difficulties. One problem is that the dimension of $\boldsymbol{\theta}$ usually grows much faster than that of $\mathbf{F}$. For instance, given a 10-station 2-D assembly process, the dimension of $\boldsymbol{\theta}$ will be 40 (vs. 16 in the four-station process in Fig. 1), whereas the dimension of $\mathbf{F}$ could very well remain 8. For a design space with a high dimension, the efficiency of the abovementioned methods is problematic. Another problem is related to the optimization of the predictive model. Suppose that we had a predictive model $\tilde{S}(\boldsymbol{\theta})$ that represents the true surface of $S(\boldsymbol{\theta})$ quite well. Then the $\tilde{S}(\boldsymbol{\theta})$ will be as nearly complicated as $S(\boldsymbol{\theta})$ and, as such, optimization of $\tilde{S}(\boldsymbol{\theta})$ will not be an easy job. A random search method (SA or GA), which we are trying to avoid, very likely may be necessary. Of course, there are still benefits, because $\tilde{S}(\boldsymbol{\theta})$ would be cheaper to compute than the original $S(\boldsymbol{\theta})$. However, the benefit usually hinges on how well $\tilde{S}(\boldsymbol{\theta})$ represents the true surface and how computationally expensive the original $S(\boldsymbol{\theta})$ is. For our fixture design problem, we did not find that the kriging model for $\{\boldsymbol{\theta}_i, S_i\}$ presents a clear advantage over the kriging model using feature functions.

2.7 Selection of K and J

One issue that we left out in Section 2.5 is how to select K (cluster number) and J (seed design number). The importance of these two values, K and J, is obvious because they determine the number of designs in the resulting design library. Apparently, these two factors are related to both the optimal sensitivity value that our method can find and the computation time that it consumes.

Unfortunately, a theoretical tie between the clustering result and the behavior of a response surface has not yet been established. Using the multistation fixture design at hand, we further investigate this issue using an experimental design approach. For a given combination of K and J, we choose two response variables, the smallest sensitivity value (before a local optimization method is applied) and the computation time. For this data-mining–aided optimal design, the overall computation time can be calculated by $T_0 + KJ \cdot T + N_f \cdot T$, where T_0 is the time component in addition to that for evaluating the objective function, known as the "overhead time" due to the uniform-coverage selection and clustering/classification processes. The second and third components are directly related to the times that the objective function is evaluated. For a given engineering design problem and a given choice of K and J, the first and second time components will be largely fixed, with T also a constant. Hence we use N_f as the second response variable.

Table 1. Results for Different Design Conditions

	Sensitivity value (S) J			The number of designs in the good design subset (N_f) J		
K	5	10	15	5	10	15
3	3.8870	3.8870	3.8870	1,361	186	624
	3.9082	3.8821	3.9335	592	628	138
	3.9488	3.9512	3.9024	938	497	142
6	3.9134	4.0080	3.9582	881	56	58
	3.8824	3.9082	3.8821	380	249	152
	3.9024	3.9024	3.9275	299	410	65
9	3.8870	3.8870	3.8934	258	136	46
	3.9335	3.9083	3.9082	210	47	112
	3.9024	3.9225	3.9244	315	270	53

We conduct a 3^2 factorial experiment, with three levels of K and J chosen at 3, 6, and 9 and 5, 10, and 15. We limit ourselves to the cases with $K < 9$, because a large K will easily result in a large KJ, a situation less likely to be computationally advantageous. Because of the previously mentioned random selection of the second locator on a panel, for a given combination of K and J, the sensitivity and N_f are in fact random variables. Then three replications are performed at each combined level of K and J. A total of 27 computer experiments are conducted, each of which will go through the procedure as outlined in Figure 3 (before applying the local optimization). The lowest sensitivity value and the value of N_f are recorded in Table 1.

From Table 1, we see that the sensitivity value S is not significantly affected by the choice of K and J. But the value of N_f is significantly affected, ranging from over 1,000 to less than 100, depending on the choice of K and J. An ANOVA using the S and N_f data confirms this finding. K and J are significant in the case of N_f at the 5% level, and their interaction has a p value of .067, suggesting that N_f is much more sensitive than S to variations in K and J.

The reason that a choice of K and J will not have much effect on the sensitivity value is related to the fact that N_f changes accordingly for different K and J. When K and J are small (i.e., the designs in the library are fewer), the partition of design sets corresponding to different levels of sensitivity is rough, and thus the resulting selection rule generated by the design library is not very discriminating. As a result, when the rule is applied to the entire set of design representatives, there will be a large number of designs that will satisfy it (e.g., the average of N_f is 964 for $K = 3$, $J = 5$). Evaluation of the large number of the selected designs, however, will circumvent the limitation brought about by the nondiscriminating selection rule, and the whole design process is still able to yield a low sensitivity value eventually. On the other hand, when a relatively large number of designs is chosen to constitute the design library, the resulting selection rule will be discriminating and is able to select a small number of good designs, evaluation of which will give us a comparably low sensitivity value. Apparently, the adaptive nature of N_f makes the eventual sensitivity value insensitive to the initial choice of K and J.

Therefore, the choice of K and J will mainly affect the algorithm efficiency, benchmarked by how many times the objective function is evaluated. The case with both small K and J is

not an efficient choice because of a large N_f. However, a large K and J is not a good choice either, because KJ will be large even though N_f will decrease. Define the total number of function evaluations as $N_t \equiv KJ + N_f$. Using the data in Table 1, we can fit a second-order polynomial, expressing N_t in terms of K and J as

$$\hat{N}_t = 2{,}220.1 - 254.0 \cdot K - 163.9 \cdot J + 8.9 \cdot KJ + 9.0 \cdot K^2 + 3.7 \cdot J^2. \quad (5)$$

Based on the foregoing expression, it is not difficult to show that the combination of $K = 8$ and $J = 13$ will give the lowest value of N_t. This combination of K and J is optimal only within the experimental range. But the benefit of a decreasing N_f does not appear to be much beyond the point of $K = 9$ and $J = 15$, where $KJ = 135$ is already more than the average value of N_f (which is 70). Any further increase in KJ is likely to outnumber the decrease in N_f.

Using the following approximation, we provide a guideline for choosing K and J, which is independent of the specific relation in (5). Recall that the good design subset is generated by passing N_r design representatives through the design selection rule. To have a meaningful design selection rule, the corresponding end node in the classification tree must have at least one design point. Suppose that there is only one design in the end node. Then the percentage of good designs selected from KJ designs in the library is $1/KJ$. If the same percentage applies to all of the entire design representatives, then $N_f = N_r/KJ$. The total number of function evaluations can be approximated as

$$N_t \approx KJ + \frac{N_r}{KJ}. \quad (6)$$

The foregoing equation suggests that N_t is minimized when $KJ = \sqrt{N_r}$. In our problem, given $N_r = 3{,}496$, KJ is roughly 60. A reasonable choice of K and J would be $K = 6$ and $J = 10$.

In actual cases, a classification tree pruned by cross-validation usually keeps more than one element in its end nodes. We also observe that the percentage of good designs selected from the design representatives may be higher than that from the design library. These factors make the actual value of KJ minimizing N_t larger than what is estimated from (6). We could treat $KJ = \sqrt{N_r}$ as the lower bound for choosing K and J. As a rule of thumb, we recommend choosing the cluster number K from 6 to 9 and the number of seed designs J per cluster from 10 to 15.

Decisions regarding cluster number is a major research topic in statistics. Tibshirani, Walther, and Hastie (2001) proposed a gap statistic for determining cluster number and also provided a comparison of several available statistical rules, including Milligan's method, Krzanowski's method, Hartigan's method, Kaufman's silhouette statistics, and their own gap statistic method. (For the details of these criteria and computational procedures, see Tibshirani et al. 2001 and references therein.) Using these criteria for our fixture design problem, the cluster number selected ranges from 2 to 5, as shown in Table 2. According to our previous discussion, these resulting cluster numbers appear to be too small and will likely cause a large N_f. Because those criteria are originally devised for a different purpose, it is not really surprising that directly applying them here may not serve our optimal design well enough.

Table 2. The Number of Clusters Suggested by the Other Methods

	Milligan's method	*Krzanowski's method*	*Hartigan's method*	*Kaufman's silhouette statistics*	*Tibshirani's gap statistic*
K	5	5	2	3	3

2.8 Overall Description of the Proposed Method

Finally, we provide a generic description of the proposed method as follows:

A. Choose a collection of N_r designs Δ that are evenly spread out in the design space using the method described in Section 2.3.

B. Identify a relatively small number of feature functions $F_1, \ldots, F_f$ that are believed to be related to the objective function and are easy to compute.

C. For each design in Δ, compute the corresponding values $\mathbf{F} = (F_1, \ldots, F_f)$ of the feature functions.

D. Cluster the values $\mathbf{F}$ of these feature functions into K clusters using K-means clustering.

E. For each of these K clusters, identify the J designs that correspond to the J values of $\mathbf{F}$ that are closest to the center of the cluster. This is the design library. Note that KJ should be $\ll N_r$.

F. For each of the KJ designs in the design library, compute the objective function S.

G. Using the KJ values of $\mathbf{F}$ and S corresponding to the designs in the design library, use CART to classify $F_1, \ldots, F_f$ into values leading to "optimum" values of S.

H. Compute the values of $\mathbf{F}$ for all designs in Δ and apply the classification rule resulting from CART to these values. The result is a small subset δ of Δ that contains designs likely to produce "optimum" values of S.

I. Apply some local optimization algorithm to the designs in δ to determine the one that optimizes S.

J. As an alternative to steps G, H, and I, instead of fitting a CART model to $\{\mathbf{F}, S\}$ in the design library, fit a MART or a kriging model. Then use the MART or kriging model to select the best design in Δ. Some local optimization algorithm can be applied to the selected design to further improve the "optimal" solution.

3. PERFORMANCE COMPARISON AND DISCUSSION

In this section we compare the performance of our data-mining–aided optimal design algorithm (before a local optimization is applied) with other optimization routines. Our design algorithm is implemented with the choice of $K = 8$ and $J = 13$, the optimal combination found in Section 2.7. Three versions of the data-mining aided optimal design are realized, using a CART, an MART, and a kriging model.

The other optimization algorithms in this comparison include simulated annealing, simplex search, and a direct evaluation of all design representatives selected by the uniform coverage selection procedure described in Section 2.3. The performance of a simulated annealing is largely determined by parameter $k_B \in [0, 1]$, known as the Boltzmann's constant. A larger k_B

(close to 1) will cause the algorithm to converge quickly, but probably to a local optimum. In contrast, a smaller k_B will aid in finding the global optimum or a solution closer to the global optimum, but at the cost of a long computing time. A general guideline given by Viswanadham, Sharma, and Taneja (1996) is to choose k_B between .85 and .95. We include the cases with $k_B = .9$ and $k_B = .95$ in our comparison.

The performance indices for comparison include the lowest sensitivity value that an algorithm can find and the time that it consumes. The objective function for the assembly process in Figure 1 is not really an expensive one, because of various simplifications that we made in variation modeling (e.g., a 2-D assembly, rigid part assumption, only four stations); the T is only .018 seconds on a computer with a 2.20-GHz P4 processor. In this study we purposely use this objective function so that we can afford to perform the explorations in Section 2. When a computationally inexpensive function is used, the overhead computing cost T_0 kicks in, which may blind us to the benefit of the proposed method for a complicated system with a more expensive objective function. To show the potential benefit for expensive objective functions, we also include for comparison the number of times that the objective function is evaluated; when T is large, the time of function evaluation essentially dominates the entire computation cost.

We implemented the aforementioned optimization algorithms to solve the multistation fixture-layout design in the MATLAB environment; for example, the MATLAB function "*kmeans*" is used for the K-means clustering method, the functions "*treefit*" and "*treeprune*" are used for the tree generation, and the function "*fminsearch*" is used for the simplex search. All optimization methods are executed on the same computer, and the average performance data of 10 trials are presented in Table 3. Based on the comparison, we have a few remarks:

1. The best design is found by the simulated annealing with $k_B = .9$ at the cost of 542.8 seconds of computation time, or more than 28,000 times of function evaluation. In comparison, the data-mining–aided design using CART reaches a very close sensitivity value (only 2.2% higher than what the SA found) but uses one-tenth of the computation time. We also notice that the data-mining–aided design using CART evaluates the objection function only one-hundredth the number of times that the SA did. The SA with a larger k_B is not advantageous; the computing time is still long (five times the data-mining–aided method for $k_B = .95$), but the resulting sensitivity value increases considerably.

2. Because of our current choice of objective function, the time that a data-mining–aided method takes is dominated by its overhead time, close to 50 seconds when using a CART model. Because we used scalable feature functions, these overhead time components will not change much even for a system with an expensive objective function. But computation for other algorithms, such as SA and simplex search, is mainly the result of evaluating the objective function. Therefore, the benefit of our data-mining–aided design method will be more obvious—28,503T for SA versus 192T for our method—for a larger, more complicated system, where the evaluation of the objective function will dominate the overall computation cost.

3. The solutions found using MART and the kriging model are on average the same. The optimal designs that they found are not as good as the one found by using CART followed by an evaluation of the good design subset. However, these high-predictive-power methods do not have to evaluate the extra designs in the good design subset and thus would be valuable tools if the extra saving in objective function evaluation is highly desirable. The foregoing optimal values are obtained when the same design library used for CART is used to establish the MART or the kriging model. The MART and the kriging model can usually find a better design when using a larger design library (i.e., a larger KJ). If we increase the size of the design library to the level corresponding to the combination of the design library for CART and the subsequent subset (i.e., a total 192 designs or, equivalently, $K = 12, J = 16$), then the MART or the kriging model can find designs as good as $S = 3.9468$ and $S = 3.9450$.

4. Another competitive solution for this fixture-design problem is to directly evaluate all 3,496 design representatives and select the best design among them. This provides a simple method of optimization. In the foregoing example, for instance, this direct comparison method finds the second-best design among the chosen optimization methods. The direct comparison method based on a uniform selection is also robust; that is, its performance is less sensitive to the properties of response functions, the properties of constraints, or the choice of initial conditions. The same philosophy of optimization was advocated by Fang and Wang (1994) using their NTM-based uniform number generation. The limitation of this solution is that it may need to evaluate a rather large number of design representatives and thus becomes computationally unaffordable when the objective function is expensive (3,496T vs. 192T or 105T in the case of fixture layout design). Determining how to reduce the number of function evaluations is exactly where a data-mining method can help.

In summary, the advantages of the data-mining–aided design are noteworthy. For the multistation fixture layout design, it

Table 3. Comparison of Optimization Methods

Optimization method	*S*	*Time (sec)*	*Time for evaluating the objective function*
Simplex search	6.825	73.8	3,200T
Simulated annealing ($k_B = .9$)	3.831	542.8	28,503T
Simulated annealing ($k_B = .95$)	3.979	259.5	13,606T
Direct evaluation of design representatives	3.892	79.3	3,496T
Data-mining–aided design using CART model	3.913	52.3	192T
Data-mining–aided design using MART model	4.025	35.8	105T
Data-mining–aided design using kriging model	4.020	41.8	105T

yields a solution with a sensitivity value as low as a random search method can find while taking a shorter time than a local optimization method (the simplex search takes 73.8 seconds). Local optimization can be applied to the best design found by the data-mining method. It will lead to a small improvement by reducing the sensitivity value further down to 3.868. It is our observation that the data-mining–aided design can often produce a satisfactory design result without the need to apply a local optimization method.

Regarding the design selection rule found by the CART shown in Figure 6, we find that for the feature of between-locator distance, only the extreme values (the largest one, F_1; the smallest one, F_5; and the second-smallest one, F_4) matter. In fact, the restriction on F_1 is $F_1 > 177.28$ mm, which will be satisfied in most designs. Hence more insights come from the rules associated with F_4 and F_5, which provide nontrivial conditions, leading us to a design with low sensitivity values. For the feature of distance change ratio, all three related feature functions play a role, that is, $F_6 > 1.15$, $F_7 > 1.85$, and $F_8 < 2.43$. Basically, this set of rules suggests that a good design will probably have a distance change ratio between 1.15 and 2.43. But the average ratio should be more than 1.85. The set of design selection rules makes our original intuitions about the across-station transition effect more quantitatively understandable. This understanding can be extended to facilitate the design of a larger system with many more stations.

The best fixture layout found by our data-mining–aided method is shown in Figure 7, where a fixture location is denoted by "+." The fact that both locators on the rear quarter panel are on the same side of the panel's gravity center does not cause problems here, because the panels are positioned on a horizontal platform in our application. If the panels are actually vertically positioned, then a force closure constraint in addition to the geometrical constraint $G(\cdot)$ should be included in the optimization scheme [i.e., in eq. (1)] to ensure that the resultant force and moment will be zero. In that situation, the resulting optimal fixture layout is different, but there is almost no change in the design procedure.

4. CONCLUDING REMARKS

This article presents a data-mining–aided optimal design method. The method is applied to facilitate the optimal design of fixture layout in a four-station SUV side panel assembly process. Compared with other available optimization methods, the data-mining–aided optimal design method demonstrates clear advantages in terms of both the sensitivity value that it can find (only 2.2% higher than what a SA found) and the computation time that it consumes (shorter than a simplex search and one-tenth of what a SA takes). The benefit could be more obvious for a larger system with a computationally expensive objective function. This method, although demonstrated in the specific context of fixture layout design, is actually rather flexible. It can be applied to a broad variety of objective functions and design criteria. It can also easily handle complicated geometric and physical constraints.

The reason that data-mining methods can facilitate optimal engineering design lies in its capability in knowledge discovery, knowledge transfer, and knowledge encapsulation. The clustering method actually connects, without performing direct evaluation of an objective function, vague human knowledge about an engineering system to design parameters and objectives that are mathematically defined. The reduction in evaluating an objective function will eventually generate a remarkable benefit in terms of algorithm efficiency. Meanwhile, the knowledge about the performance of an engineering system will become more explicit and numerical once the set of design selection rules is

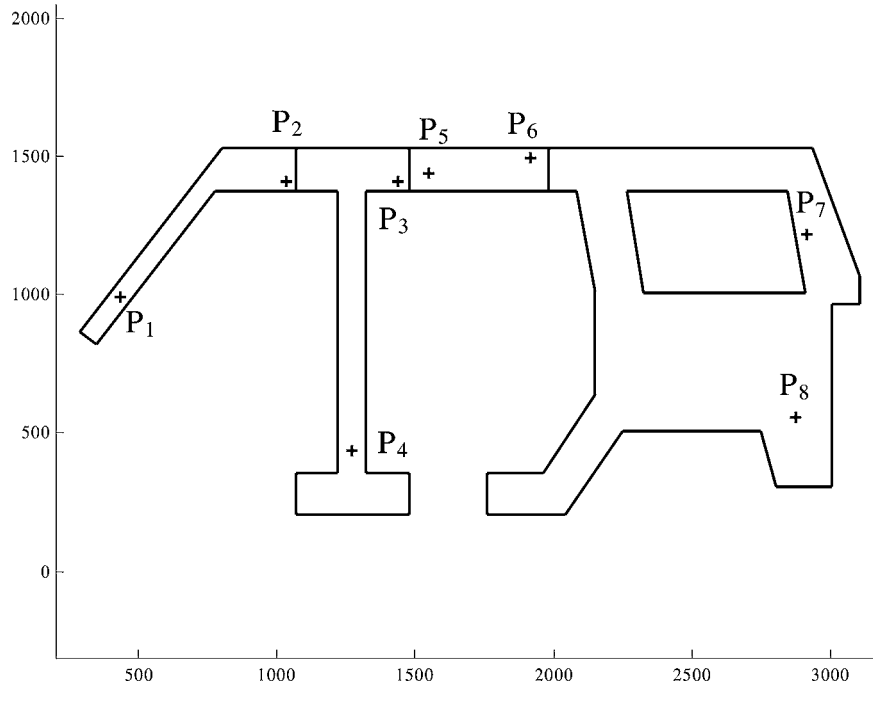

Figure 7. Optimal Fixture Layouts With the Lowest S Value.

formed from a classification method. The accumulated knowledge, expressed in design rules and the optimal design conditions, can be translated into the optimization of a similar, yet larger system.

Finally, we would like to add one more note on the use of feature functions, which transfer engineering knowledge for statistical treatments. Such an integration of engineering knowledge and statistical methods is considered an important way of improving statistical solutions when solving messy engineering problems. Traditional ways of transferring engineering knowledge include expert systems (Jackson 1999) and physical modeling. The former method is usually too qualitative, and the latter is highly quantitative but less flexible; in many sophisticated physical systems, accurate physical modeling of the system is almost impossible. We feel that the inclusion of feature function strikes a balance between being more quantitative, and also sufficiently flexible in incorporating engineering knowledge and understanding into the process of design optimization.

ACKNOWLEDGMENT

The authors gratefully acknowledge financial support from National Science Foundation grants DMI-0217481 and DMI-0348150 and from the State of Texas Advanced Technology Program grant 000512-0237-2003. The authors also thank the editor and referees for their valuable comments and suggestions.

[Received September 2003. Revised November 2004.]

REFERENCES

Bertsimas, D., and Tsitsiklis, J. (1993), "Simulated Annealing," *Statistical Science*, 8, 10–15.

Camelio, A. J., Hu, S. J., and Ceglarek, D. J. (2003), "Modeling Variation Propagation of Multi-Station Assembly Systems With Compliant Parts," *Transactions of the ASME, Journal of Mechanical Design*, 125, 673–681.

Ding, Y., Ceglarek, D., and Shi, J. (2000), "Modeling and Diagnosis of Multi-Station Manufacturing Processes: State Space Model," in *Proceedings of the 2000 Japan/USA Symposium on Flexible Automation*, July 23–26, Ann Arbor, MI, 2000JUSFA-13146.

Fang, K. T., Lin, D. K. J., Winkle, P., and Zhang, Y. (2000), "Uniform Design: Theory and Application," *Technometrics*, 42, 237–248.

Fang, K. T., and Wang, Y. (1994), *Number-Theoretic Methods in Statistics*, New York: Chapman & Hall.

Gen, M., and Cheng, R. (2000), *Genetic Algorithms and Engineering Optimization*, New York: Wiley.

Hastie, T., Tibshirani, R., and Friedman, J. (2001), *The Element of Statistical Learning*, New York: Springer-Verlag.

Hillier, F. S., and Lieberman, G. J. (2001), *Introduction to Operations Research* (7th ed.), New York: McGraw Hill.

Igusa, T., Liu, H., Schafer, B., and Naiman, D. Q. (2003), "Bayesian Classification Trees and Clustering for Rapid Generation and Selection of Design Alternatives," in *Proceedings of 2003 NSF Design, Service and Manufacturing Grantees and Research Conference*, Birmingham, AL, January 4–9.

Jackson, P. (1999), *Introduction to Expert System* (3rd ed.), Reading, MA: Addison-Wesley.

Jin, J., and Shi, J. (1999), "State-Space Modeling of Sheet Metal Assembly for Dimensional Control," *Transactions of ASME, Journal of Manufacturing Science & Engineering*, 121, 756–762.

Kim, P., and Ding, Y. (2004), "Optimal Design of Fixture Layout in Multi-Station Assembly Process," *IEEE Transactions on Automation Science and Engineering*, 1, 133–145.

McKay, M. D., Bechman, R. J., and Conover, W. J. (1979), "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code," *Technometrics*, 21, 239–245.

Nelder, J. A., and Mead, R. (1965), "A Simplex Method for Function Minimization," *The Computer Journal*, 7, 308–313.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003), *Design & Analysis of Computer Experiments*, New York: Springer-Verlag.

Schwabacher, M., Ellman, T., and Hirsh, H. (2001), "Learning to Set Up Numerical Optimizations of Engineering Designs," in *Data Mining for Design and Manufacturing*, ed. D. Braha, Boston, MA: Kluwer Academic, pp. 87–125.

Tibshirani, R., Walther, G., and Hastie, T. (2001), "Estimating the Number of Clusters in a Data Set via the Gap Statistic," *Journal of the Royal Statistical Society*, Ser. B, 63, 411–423.

Viswanadham, N., Sharma, S., and Taneja, M. (1996), "Inspection Allocation in Manufacturing Systems Using Stochastic Search Techniques," *IEEE Transactions on Systems, Man and Cybernetics—Part A*, 26, 222–230.