A Design Preference Elicitation Query as an Optimization Process

Yi Ren[*]
Ph.D. Candidate
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan 48109
Email: yiren@umich.edu

Panos Y. Papalambros
Professor, Fellow of ASME
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan 48109
Email: pyp@umich.edu

We seek to elicit individual design preferences through human-computer interaction. During an iteration of the interactive session, the computer queries the subject by presenting a set of designs from which the subject must make a choice. The computer uses this choice feedback and creates the next set of designs using knowledge accumulated from previous choices. Under the hypothesis that human responses are deterministic, we discuss how query schemes in the elicitation task can be viewed mathematically as learning or optimization algorithms. Two query schemes are defined. Query Type 1 considers the subject's binary choices as definite preferences, i.e., only preferred designs are chosen while others are skipped; Query Type 2 treats choices as comparisons among a set, i.e., preferred designs are chosen relative to those in the current set but may be dropped in future iterations. We show that Query Type 1 can be considered as an active learning problem while Type 2 as a "black-box" optimization problem. This paper concentrates on Query Type 2. Two algorithms based on support vector machine and efficient global optimization search are presented and discussed. Early user tests for vehicle exterior styling preference elicitation are also presented.

Nomenclature

$\mathbf{1}$	column vector where all elements are 1's
b	bias for a linear decision function
C	soft-margin SVM weight
$\mathcal{D}$	design space with p dimensions
$\mathcal{D}_+$	most preferred region in $\mathcal{D}$
$\mathbf{K}$	$l \times l$ kernel matrix
l	number of designs in each design set during an interaction
p	dimensionality of the design space
$\mathbf{R}$	correlation matrix in the kriging model
t	maximum number of iterations during an interaction
s^2	mean square error in a kriging model
$u(\mathbf{x})$	utility function defined on $\mathcal{D}$
$\mathbf{w}$	coefficients for a linear decision function
w_1, w_2	weights in the weighted sum merit function
$\mathbf{x}$	a design in $\mathcal{D}$
y	response for $\mathbf{x}$
$\hat{y}$	prediction of $y(\mathbf{x})$ for a certain model
α	solutions to the soft-margin SVM
$\hat{\beta}$	maximum likelihood estimation in the simple kriging model
λ^{kriging}	Gaussian spread used in $\mathbf{R}$
λ^{SVM}	Gaussian spread used in $\mathbf{K}$
ξ_i	classification error for sample i

1 Introduction

The present work is motivated by the observation that while consumer preference plays an essential role in product design, its effective integration within design optimization remains challenging. Research in linking preference models to engineering optimization models for consumer product design (e.g., Wassenaar et al. [1], Kumar et al. [2], Li and Azarm [3], Michalek et al. [4]) has demonstrated how these different disciplines can be integrated. Most of these demonstrations utilize preference models with only a small number of variables since the variance in coefficient estimation of these models can be large and unfavorable when the dimensionality of the design space becomes high. Yet, to cap-

[*]Corresponding author.

ture the right design variables in an essentially qualitative or holistic design problem such as vehicle styling we must have sufficient design freedom and thus a high-dimensional design space. This "curse of dimensionality" is alleviated by static design of experiments (e.g., Kuhfeld et al. (1994) [5], Hoyle et al. (2009) [6] on D-efficient sampling) as well as adaptive methods (e.g., Toubia et al (2004) [7], Abernethy et al. (2007) on adaptive sampling [8]). However, these methods are all model-based, i.e., tests are designed so that they can estimate effectively the preference models. Utility models are prevalent in preference research, but the validity of such models is still under discussion, see Netzer et al. 2008 [9].

An alternative way of capturing preferences without a preference model is through the use of human-computer interactions where the computer gradually refines its assessment of people's preferences based purely on user feedback. Interactive evolutionary computation (IEC) [10] falls into this category; a user evaluates design fitness and the evolutionary scheme in then executed as in normal evolutionary computation (e.g., Tokui et al. and Johanson et al. on music design [11] [12], Kim et al. on dressing design [13], Kelly et al. on vehicle silhouette design [14]). Two drawbacks of IEC are that (i) the fitness requests force users to assign values for each individual design, which is not a natural way for people to express preferences and can cause user fatigue and hamper convergence [10]; (ii) designs close to not-preferred designs presented earlier are likely to appear in later iterations due to the stochastic nature of IEC schemes.

In this paper, we introduce an interaction similar to those in IEC but relieving the burden on users by requiring only binary feedback. Users assign only "preferred" and "not-preferred" labels to designs. The research goal is to investigate how a search algorithm should be designed to elicit effectively the most preferred designs using this binary information with a tolerable interaction effort, namely, how to converge with a small number of iterations. Like with IEC, the focus here is not preference modeling and estimation but structuring the interaction to speed up convergence with limited elicitation (data collection).

1.1 Problem Formulation

To start, we formulate the preference elicitation process as an optimization problem by assuming that users can evaluate designs with their own deterministic preference function. The assumption we make here requires further scrutiny for at least two reasons: (1) Using functions to model preferences implies transitivity; however, research (e.g., Petiot et al. (2006) [15]) shows that such transitivity may not hold, i.e., one can prefer A over B and B over C, but also C over A; (2) even if transitivity holds, the function may not be deterministic during the interaction; as investigated in MacDonald et al.(2008) [16], people may construct preference on an as-needed basis and the underlying preference model may change along with the change of the designs presented. While with caution, we use this assumption based on the reported success in both marketing and IEC research.

Next, we look at the forms of the preference function. We examine two types of queries that can be presented to the subject. We will show that user feedback from these queries should be interpreted differently, leading to different forms of preference function and thus different search strategies. For clarification, a few definitions are first introduced below. Throughout the paper we use the terms "human", "user", "subject" or "consumer" interchangeably, as preference elicitation can be used in any of those contexts within the design process.

We formulate the interaction problem as follows. Let $\mathcal{D} \subset \mathbb{R}^p$ be the design space where any $\mathbf{x} \in \mathcal{D}$ represents a design described with p variables. Assume that the user is presented with l design alternatives in each iteration and is asked to select any number of designs as preferred and the rest as not preferred. We represent this user response with a function $f(\mathbf{x}_1, \{\mathbf{x}_i\}_{i=2}^l) : (\mathcal{D}, \mathcal{D}^{l-1}) \to [0,1]$ that maps the design $\mathbf{x}_1$, with respect to a fixed set of $l-1$ designs $\{\mathbf{x}_i\}_{i=2}^l$, to a 0-1 value, where 1 represents a preferred design and 0 represents a not-preferred one. The user is allowed to chose none of the presented designs. Our objective is to locate a "most preferred" region in the design space $\mathcal{D}$ defined as follows.

Definition 1. *A design $\mathbf{x}_1$ belongs to the most preferred region $\mathcal{D}_+ \subseteq \mathcal{D}$ if and only if $\forall \{\mathbf{x}_i\}_{i=2}^l \in \mathcal{D}^{l-1}$ and $\forall l = 2,3,...,$ we have $f(\mathbf{x}_1, \{\mathbf{x}_i\}_{i=2}^l) = 1$.*

Note that the l counter starts at 2 because we must have at least two designs to enable comparison.

Let us now discuss two possible types of questions we can use during a user-computer interaction. As noted later in this paper, the difference between these two questions may be subtle but important. The first question is: "Is there a preferred design in the presented set?". The user is asked to pick only preferred designs rather than comparing the designs in the set. Thus, a design chosen as preferred will always remain so. We call this Query Type 1. Mathematically, this means that the response function can be modeled as an indicator function defined on $\mathcal{D}$ and independent of the parameter set $\{\mathbf{x}_i\}_{i=1}^{l-1}$, as given in Definition 2:

Definition 2. *User response is the indicator function:*

$$f(\mathbf{x}, \cdot) := 1_{\mathcal{A}}(\mathbf{x}) = \begin{cases} 1, \ \textit{if}\ \mathbf{x} \in \mathcal{A}, \\ 0, \ \textit{if}\ \mathbf{x} \notin \mathcal{A}. \end{cases} \quad (1)$$

The definition implies that each design within $\mathcal{D}$ is associated with a definite label and is independent of other accompanying designs. By Definition 1, $\mathcal{A} = \mathcal{D}_+$. Thus, learning user preference can be treated as a classification problem to approximate $\mathcal{D}_+$ by querying labels of sample points in $\mathcal{D}$. The methodology for solving this problem exists in statistical learning. The term "active learning" was first coined by Tong and Koller (2001) who showed that faster classification can be achieved by querying samples iteratively [17]. By formulating the classification task using Support Vector Machine (SVM) (see Vapnik 1998 [18]), they proved that efficient queries can be made by cutting the space of the classifier coefficients into equal halves. Further research discussed the balance between exploration (querying without using knowledge gained) and exploitation (like Tong and Koller, querying with all knowledge gained) to achieve more stable performance on different problems (Osugi 2005 [19] and Baram 2004 [20]). A "locking" problem in using active learning for adaptive sampling in a continuous space was reported by Basudhar et al. [21] and a heuristic sampling method was proposed to overcome it.

A drawback of Query Type 1 is that it prevents users from selecting relatively better designs in the presented set when none of them is close to their preference. Query Type 2 thus comes into play, namely, "Which designs in the set do you prefer more?". Introducing an arbitrary utility function $u(\mathbf{x}) : \mathcal{D} \to \mathbb{R}$, a design $\mathbf{x}_1$ is referred to be better than $\mathbf{x}_2$ if and only if $u(\mathbf{x}_1) > u(\mathbf{x}_2)$. Further, to transform the real-valued utilities to binary choices, we introduce a mapping M: $\mathbb{R}^l \to \{0,1\}^l$ that clusters a set of l designs to binary classes. The response function for this query type can be then modeled as follows.

Definition 3. *A response function f with utility and clustering is defined as*

$$f(\mathbf{x}_1, \{\mathbf{x}_i\}_{i=2}^{l}) = M(\{u(\mathbf{x}_i)\}_{i=1}^{l})_1. \tag{2}$$

If the utility function is continuous and bounded on $\mathcal{D}$ and if there always exists a zero in the set $M(\{u(\mathbf{x}_i)\}_{i=1}^{l})$, i.e, a not-preferred design can always be distinguished, then $\mathcal{D}_+$ only contains $arg\max_{\mathbf{x}}\ u(\mathbf{x})$. Indeed, if there exists $\mathbf{x} \in \mathcal{D}_+$ and $\mathbf{x} \neq arg\max_{\mathbf{x}}\ u(\mathbf{x})$, then we can always find a series $\{\mathbf{x}_i\}$ such that $u(\mathbf{x}_i) > u(\mathbf{x})$, $\forall i$. Thus, $\mathbf{x}$ will be assigned a zero and $\mathbf{x} \notin \mathcal{D}_+$. With these modeling assumptions, Query Type 2 corresponds to a "black-box" optimization problem with binary outputs.

In the present paper, we focus on search algorithms associated with Query Type 2 and leave Query Type 1 and active learning algorithms for future research reporting. We present two search algorithms: (1) SVM-Search, a heuristic method that shrinks the search space by intersecting decision boundaries from different iterations; (2) EGO search, a modification of the well-established efficient global optimization (EGO) algorithm that samples the design space based on the response surface of current observations. Results with simulated user interactions show that with only binary feedback available, the SVM search algorithm is more effective than a genetic algorithm, while EGO search is better than both. Therefore, the EGO search algorithm is implemented in actual user tests for eliciting preferences in vehicle exterior styling design, as discussed in Section 5.

The remainder of the paper is arranged as follows: In Section 2 we review briefly SVM and EGO; in Sections 3 and 4 we describe the SVM and EGO search algorithms, respectively, developed for Query Type 2, and discuss their performance with simulated user tests; in Section 5 we implement the EGO search for a three-dimensional vehicle styling design problem, present some early user tests, and discuss algorithm convergence and some practical issues; we conclude in Section 6.

2 Background

This section outlines the basic ideas in SVM and EGO.

2.1 Support Vector Machine

The formulation of SVM as a classification tool results directly from the Vapnik − Chervonenkis (VC) theory which states that to minimize the prediction error, a balance is needed between minimizing the training error and controlling the model complexity [18]. This motivates the formulation of SVM for binary classes (soft-margin SVM) based on the observations $\{\mathbf{x_i}\}_{i=1}^{l}$ and their labels $\{y_i\}_{i=1}^{l} \in \{-1,1\}^l$. In the numerical setup of an SVM problem, we use binary labels $\{-1,1\}$ rather than $\{0,1\}$.

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \mathbf{w}^T\mathbf{w} + C\sum_{i}^{l}\xi_i, \\ s.t. \quad & y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1,...,l. \end{aligned} \tag{3}$$

The solution to the problem in Eqn. (3) finds a linear decision function $\hat{y} = \mathbf{w}^T\mathbf{x} + b$ that minimizes a merit function which combines both the training error $\sum_i^l \xi_i$ and the model complexity $\mathbf{w}^T\mathbf{w}$ with the parameter C. The resulting hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ separates the two classes and is called the decision boundary. Further, since the solution of Eqn. (3) depends only on the similarity matrix $K_{ij} = \mathbf{x}_i^T\mathbf{x}_j$, $\forall i, j = 1,...,l$, the linear classifier $y = \text{sign}(\mathbf{w}^T\mathbf{x}+b)$ can be replaced by any nonlinear one with a proper definition of the similarity $K_{ij} = \psi(\mathbf{x}_i)^T\psi(\mathbf{x}_j)$, where $\psi(\cdot) : \mathcal{D} \to \mathcal{F}$ represents any mapping of $\mathbf{x}$ to an unknown feature space $\mathcal{F}$. Such a method is referred to as a kernel trick. The solution for Eqn. (3), with a Gaussian kernel $K_{ij} = \exp(-\lambda^{\text{SVM}}||\mathbf{x}_i - \mathbf{x}_j||_2^2)$, can be summarized as:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{l} y_i\alpha_i \exp(-\lambda^{\text{SVM}}||\mathbf{x} - \mathbf{x}_i||_2^2) + b, \tag{4}$$

where α_i and b are solutions to the dual problem of Eqn. (3). Throughout this paper, the Gaussian (or radial basis) kernel is used with a default spread $1/p$, and the LIBSVM package [22] is used to solve the dual problem.

2.2 Efficient Global Optimization

EGO [23] is a global optimization method designed for black-box objectives and constraints. In a minimization problem, EGO queries a point that has low predicted objective value and high variance in the prediction. It is shown that such a point has the highest expected improvement in

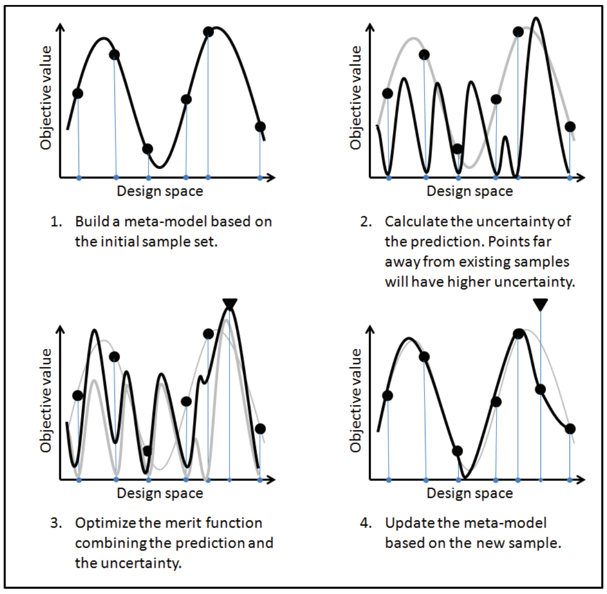

Objective value
Design space
1. Build a meta-model based on the initial sample set.
2. Calculate the uncertainty of the prediction. Points far away from existing samples will have higher uncertainty.
3. Optimize the merit function combining the prediction and the uncertainty.
4. Update the meta-model based on the new sample.

Fig. 1. EGO operation concept

minimizing the objective. Fig. 1 illustrates how EGO works. Refer to Jones et al. (1998) [23] and Sacks et al. (1989) [24] for more details.

Denoting $\{\mathbf{x}_i\}_{i=1}^{l}$ as a set of l observations and $\{y_i\}_{i=1}^{l}$ as their responses, EGO estimates a kriging (generalized linear) model to predict the response $y(x)$ as

$$\begin{aligned}
\hat{y}(x) &= \hat{\beta} + \mathbf{r}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\beta}),\\
\text{where:}&\\
r_i(x) &= \exp\left(-\lambda^{\text{kriging}}||\mathbf{x}_i - \mathbf{x}||_2^2\right),\ \forall i = 1,...,l,\\
R_{ij} &= \exp\left(-\lambda^{\text{kriging}}||\mathbf{x}_i - \mathbf{x}_j||_2^2\right),\ \forall i,j = 1,...,l,\\
\hat{\beta} &= \frac{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}. \qquad (5)
\end{aligned}$$

The forms of Eqn. (4) and Eqn. (5) are the same. While the kriging model regresses real-valued observations, the SVM model regresses binary ones. In the proposed EGO Search algorithm further below where only binary observations are available, we replace the kriging model with an SVM one.

The variance of the difference between the prediction $\hat{y}$ and the true realization y can be derived following the model estimations above as:

$$s^2(\mathbf{x}) = [1 - \mathbf{r}^T\mathbf{R}^{-1}\mathbf{r} + \frac{(1-\mathbf{1}^T\mathbf{R}^{-1}\mathbf{r})^2}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}] \qquad (6)$$

One can consider this variance as a continuous minimum distance function, i.e., $s^2(x)$ is high when x is not surrounded by existing samples and vice versa. With $\hat{y}$ and s^2 available, EGO optimizes a merit function that evaluates the expected improvement in the objective and its optimum is the next query point, namely, the point most likely to decrease the observed objective value further.

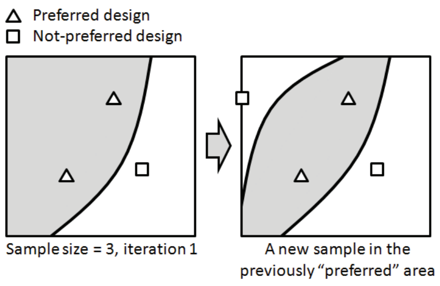

△ Preferred design
□ Not-preferred design
Sample size = 3, iteration 1
A new sample in the previously "preferred" area

Fig. 2. SVM Search scattering and classification

3 SVM Search Algorithm

SVM Search is a heuristic algorithm that aims to shrink the design space during the interaction and thus force it to converge in a few iterations. The algorithm attempts to address the problem that, when the dimensionality of the design space becomes high, search convergence can become too slow for practical user-computer interaction.

The proposed algorithm works as follows. Let the initial design space be $\mathcal{D}_0$ and that of iteration i be $\mathcal{D}_i$, where $i = 1,...,k$. The algorithm starts with l_1 random samples in $\mathcal{D}_0$. A decision boundary $g_1(\mathbf{x}) = 0$ is generated from the data set $\{\mathbf{x}_i, y_i\}_{l_1}$ once user feedback is collected. Suppose that m out of l_1 samples are labeled as $y = 1$; then $l_1 - m$ samples are drawn from the region $g_1(\mathbf{x}) > 0$. Each sample maximizes the minimum distance from all existing samples. The user is asked to label the combined set of these new samples and the m "winners" from the first stage. Once new labels are assigned and old labels are updated, a new decision boundary $g_2(\mathbf{x}) = 0$ is generated upon the current total data set $\{\mathbf{x}_i, y_i\}_{2l_1-m}$.

We illustrate this process in Fig. 2. Samples with label "0" need no further update. To speed up convergence, the decision boundary is recorded after p iterations, where p is an adjustable parameter. Any further decision boundaries drawn are combined with the recorded one in determining the future sampling space. The method can thus be considered as a way to balance exploitation (small p) and exploration (large p). This space reduction procedure is illustrated in Fig. 3. The algorithm terminates when the maximum number of iterations is reached or the design space constrained by the decision boundaries is too small to contain random samples. The pseudo code of this algorithm is shown in Fig. 4.

The "Scatter" subroutine (Fig. 5) generates l samples in the design space constrained by the decision boundary set. Each sample is generated so that its minimum distance to all of the other labeled data is maximized. The subroutine "Label" refers to human evaluations for actual user tests but is regarded as a function call in the simulated interactions tests that use known mathematical functions; it clusters objective values of the sample set using K-means and labels them as

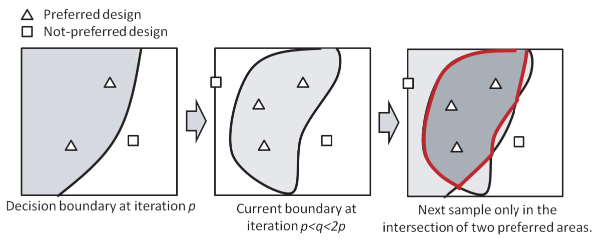

△ Preferred design
□ Not-preferred design
Decision boundary at iteration p
Current boundary at iteration p<q<2p
Next sample only in the intersection of two preferred areas.

Fig. 3. SVM Search design space reduction

```
Algorithm 1: SVM Search algorithm
input : Design space $\mathcal{D}$, max_iteration $t$,
        max_sample $l$
output: Decision boundary set $\{g(\mathbf{x}) = 0\}$
Initialization;
$g\_set \leftarrow \{\}$; // initialize decision boundary set
$\{\mathbf{x}_j\}_l =$ Scatter ($g\_set, \mathcal{D}, l$); // scatter l samples
  in D constrained by g_set
$\{y_j\}_l =$ Label ($\{\mathbf{x}_j\}_l$); // label the samples
$rec = 0$; // counter for recording decision boundaries
for $i = 2$ to $t$ do
    Check if it is time to record current decision boundaries;
    if $rec == p$ then $g\_set = g_{i-1}(\mathbf{x})$, $rec = 0$;
    else $rec = rec + 1$;
    Check termination criteria;
    $g_i(\mathbf{x})$ = SVM ($\{\mathbf{x}_j\}_l$, $\{y_j\}_l$, $svm\_options$);
    // SVM Training
    $\{\mathbf{x}_j\}_l$ = Scatter ($\{g\_set, g_i(\mathbf{x})\}, D, l - m, \mathbf{X}$);
    // Scatter samples assuming m winners in the
      previous iteration, X is the total labeled sample
      set
    $\{y_j\}_l$ = Label ($\{\mathbf{x}_j\}_l$); // Update labels
end
```

Fig. 4. Proposed SVM Search algorithm

two classes. The subroutine "SVM" calls the LIBSVM package [22], and *svm_options* stores parameters required by the package. In this study, the soft-margin SVM weight C is set at 10^6 and no parameter tuning is performed.

We compare results from simulated tests using SVM Search and the Matlab GA toolbox with binary fitness. Results from a random sampling scheme are also presented as a baseline performance. This latter algorithm merely samples points randomly during the process, keeping better points and dropping worse points. Some standard test functions, namely, 2D Rosenbrock, Six-hump Camelback, and Branin (see [25] and Eqn. (7) through (9) below), are used to simulate an interaction where the optimum of the function is the preference the user has in mind. To measure algorithm performance, we calculate the minimum gap between the global optimum of a test function and the function values at the samples. We use this form of error because in real user interactions, it is always possible to ask the question: "Among all the preferred designs, which ones are the most preferred?". Due to the stochastic nature of all search algorithms, the performance is reported as the mean and standard deviation of the minimum gaps from ten separate runs. Table 1 compares the performance of the three algorithms under maximum iteration number of 20 and maximum sampling size of 3 per iteration. The results suggest that the proposed SVM search algorithm has the best overall performance.

```
Algorithm 2: Scatter sub-routine for SVM Search
input : Decision function set $g\_set$, Design space $\mathcal{D}$,
        num_sample $l - m$, total labeled set $\mathbf{X}$
output: New sample set $\{\mathbf{x}'_j\}_l$
while $||\{\mathbf{x}'_j\}|| < l - m$ do
    $\mathbf{x}' =$ Maximin ($\mathcal{D}$, $g\_set$, $\mathbf{X}$); // x' has the
      maximum minimal distance to X in D and g_set
    $\mathbf{X} = \{\mathbf{X}, \mathbf{x}'\}$; // Update the total sample set
end
```

Fig. 5. SVM Search algorithm: scatter subroutine

Table 1. Means and standard deviations of the final error from SVM Search, GA and Random Search

Mean(Std)	Rosenbrock	Camelback	Branin
SVM Search	4.12 (6.77)	0.19 (0.33)	1.40 (2.53)
GA	4.31 (3.76)	1.02 (0.83)	31.0 (7.45)
Random	19.8 (13.5)	18.3 (18.4)	23.0 (21.7)

2D Rosenbrock function:

$$f(x,y) = -\left((1-x)^2 + 100(y-x^2)^2\right), \quad \mathcal{D} = \{(x,y),\ x \in [-1.5, 1.5],\ y \in [-0.5, 1.5]\}. \tag{7}$$

2D Six-hump Camelback function:

$$f(x,y) = -\left((4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2\right), \quad \mathcal{D} = \{(x,y),\ x \in [-3, 2],\ y \in [-3, 2]\}. \tag{8}$$

2D Branin function:

$$f(x,y) = -\left((y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x) + 10\right), \quad \mathcal{D} = \{(x,y),\ x \in [-5, 10],\ y \in [-5, 10]\}. \tag{9}$$

4 EGO Search Algorithm

A search algorithm for user-computer interaction with binary choice feedback can be built in the same fashion as

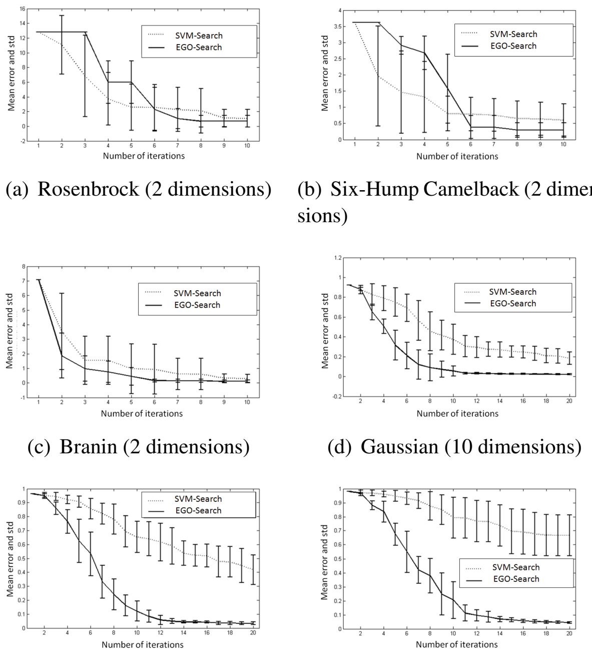
```
Algorithm 3: EGO Search Algorithm
input  : Design space $\mathcal{D}$, max_iteration $t$,
         max_sample $l$
output: Optimal solution $\mathbf{x}^*$
// Initialize data set
$\mathbf{X} = \emptyset$; $\mathbf{y} = \emptyset$;
// Initialize a Latin Hypercube sample set of size l
$\{\mathbf{x}_j\}_{j=1}^{l}$ = LQScatter $(\mathcal{D}, l)$;
// Request user feedback
$\{y_j\}_{j=1}^{l}$ = Label $(\{\mathbf{x}_j\}_{j=1}^{l})$;
// Record the initial observation
$\mathbf{X} \leftarrow \{\mathbf{x}_j\}_{j=1}^{l}$; $\mathbf{y} \leftarrow \{y_j\}_{j=1}^{l}$;
for $i = 2$ to $t$ do
    // Train accumulated data to get the decision
       function
    $g(\mathbf{x}) \leftarrow$ Train $(\mathbf{X}, \mathbf{y}, svm\_options)$;
    $m$ = number of preferred designs in the previous
    round;
    // Keep the preferred designs from the previous
       round in the sample set
    for $j = 1$ to $m$ do
        $\mathbf{x}_j$ = $j$th preferred design from round $i-1$;
    end
    // Scatter $l-m$ new sample points to maximize
       the merit function
    for $j = m$ to $l$ do
        Find $\mathbf{x}_j$ that maximizes the merit function
        $f_{\text{merit}}(g(\mathbf{x}), \mathbf{X}, \mathcal{D})$; add $\mathbf{x}_j$ to $\mathbf{X}$;
    end
    // Request user feedback
    $\{y_j\}_{j=1}^{l}$ = Label $(\{\mathbf{x}_j\}_{j=1}^{l})$;
    Update the last $l$ labels of $\mathbf{y}$ according to $\{y_j\}_{j=1}^{l}$;
end
```

Fig. 6. Proposed EGO Search algorithm

EGO. In the first iteration, the computer presents a sample set based on a Latin Hypercube design. The user then evaluates these samples and chooses the relatively preferred ones (those with relatively higher utilities). The computer takes in these binary choices as labels on the sample set. SVM is run to find the optimal decision function. A merit function combining this decision function and the mean square error function is then optimized, and its solutions along with the preferred samples from the previous round are used as the samples for the next iteration. The algorithm continues until the sample set becomes identical to the user selection or the maximum number of iteration is reached. Fig. 6 provides a summary of the steps.

Several points need to be discussed: (1) The merit function we use has the form $f_{\text{merit}}(\mathbf{x}) = -(w_1\hat{y} + w_2 s^2)$, where $\hat{y}$ is the predicted utility and s^2 is the mean square error. The weight w_1 is set to 1 and w_2 is decreasing along the iteration as $t/i - 1$, where t is the maximum iteration number and i is the current iteration number. This value is empirically set to make the two function values of the same scale, and to start the search with an emphasis on exploration and then shift priority to exploitation of the decision function towards the end; (2) when calculating s^2, the Gaussian spread parameter λ^{kriging} is set to 10; (3) the weight C in the soft-margin SVM is set at a high value of 10^6 since we consider all user input to be correct and little training error is allowed; (4) both $\hat{y}$ and s^2 are non-convex functions of $\mathbf{x}$ and thus optimizing the merit function can be costly. Jones et al. proposed a branch and bound algorithm that searches blocks in $\mathcal{D}$ with high upper bounds for both $\hat{y}$ and s^2 [23]. It is not clear whether this method will be effective when the dimensionality of $\mathcal{D}$ is high, and further study is required. As an expedient alternative, the Matlab GA toolbox [26] is used to optimize the merit function in the present study.

Fig. 7 shows results on test functions using EGO Search and SVM Search following the same test setup as in Section 3. For each algorithm and each function, ten tests are executed and the error is calculated as the current minimum gap from the theoretical optimum. The solid line and bars show the mean error and its standard deviation for the EGO Search, respectively, while the dotted ones show those for the SVM Search algorithm. EGO Search has more reliable performance overall than SVM Search especially for higher dimensions.



(a) Rosenbrock (2 dimensions) (b) Six-Hump Camelback (2 dimensions)

(c) Branin (2 dimensions) (d) Gaussian (10 dimensions)

(e) Gaussian (13 dimensions) (f) Gaussian (15 dimensions)

Fig. 7. Comparison between EGO Search and SVM Search.

5 Vehicle Exterior Styling Design Elicitation

Due to its observed performance in simulation, EGO Search is chosen for implementation in an actual user study involving vehicle exterior styling design. The purpose of the study is to investigate whether the algorithm can successfully help the subjects to search for a design that is close to a given target.

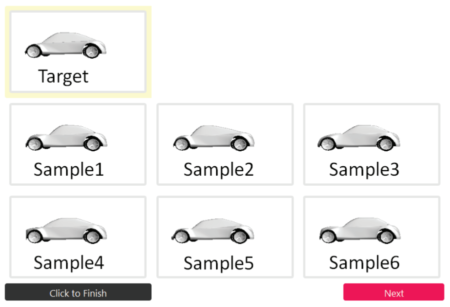
5.1 Software Development

To enable real-time online human-computer interactions on designing vehicle styles, a parametric three-dimensional (3D) model incorporating 20 variables within the range of $[0,1]$ is programmed in WebGL, which is a context of the canvas HTML element that provides a 3D graphics API implemented in a web browser without the use of plug-ins. A Java servelet implementing the proposed search algorithm runs on the server side to parse user feedback and provide queries for the next iteration. At the time of this writing, readers may access "http://yirenumich.appspot.com" for the EGO Search implementation. Firefox 5 and Google Chrome 10 (or later) are required to run these applications.

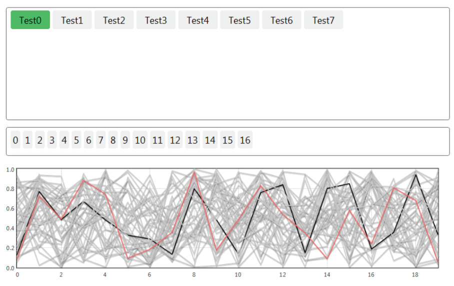
5.2 Convergence Test Setup

The test pages are set up at "/convergencetest.html" on the website and all user data are submitted and can be retrieved at "/log.html". In the test environment, a random and a fixed target design are shown at the top of the web page. Six initial random guesses are shown below the target, see Fig. 8(a). The data retrieval page shows in the top division how many tests have been conducted, followed by the accumulated data from each iteration in that test. Each design sampled thus far is visualized in the bottom division where the preferred ones are highlighted. The red highlighted curve represents the target in that test. Fig. 8(b) contains the data visualization. The parameters set up in the implementation are: $\lambda^{SVM} = \lambda^{kriging} = 0.05$ and $[w_1, w_2] = [1, 1000]$. These parameters are set empirically and their value should be further studied. Since Google Appengine limits the request time at 30 seconds, we set the population limit to 10 and the number of generations to 500 in the GA implementation for the global search inside the EGO Search iterations. This is a fairly low-cost setup for a 20-dimensional space.

This online environment was developed for mass data collection. As of this writing, more than six hundred anonymous interactions have been recorded. However, looking at the collected data, we surmised that the anonymous users do not necessarily follow the test instructions and thus, while interesting, the collected results cannot be analyzed with confidence. Consequently, we conducted a pilot test in an interview setting with eight subjects from the University of Michigan. Prior to initiating the test, the subjects were instructed on how they can view, rotate, zoom, pan, synchronize and reset the viewpoint of each design rendering. They were also made clearly aware of their task which was to pick a set of designs that were relatively closer to the given target in each iteration. Moreover, the subjects understood that they could pick up to five designs as "preferred" and that they had the option to pick nothing if none was relatively close to the target. After the subjects submitted their final design, we ask them to state whether they were "satisfied" or "not satisfied"



(a) Test environment



(b) Test data visualization

Fig. 8. Online human-computer interaction interface and data visualization. (a): The interactive environment at "/convergencetest.html" allows the user to zoom, pan, rotate each design and updates the guesses once the user hits the "Next" button; (b): The data visualization window at "/log.html" has all user tests listed at the top, number of iterations in the middle, and the cumulated data at the bottom. The red curve represents the target design, the highlighted dark curve(s) represent the preferred design at this point and the rest all not preferred.

with the test result. This information was also sent back to the server.

5.3 User Data Analysis

While all users were satisfied with the test result according to their responses, we need to examine whether the interactions converged. We start by measuring the Euclidean distance between each sampled design and the target in each test, as illustrated in Fig. 9, since no explicit objective function is available and the gap between the optimum and the objective value at each sample cannot be measured. In the figure, samples towards the end of each test curve are those generated in later iterations. The plot shows that later samples do not necessarily acquire lower distance values, and thus convergence is not observed.

Nonetheless, visual comparison between targets and user interaction results, as shown in Fig. 10, indicates that users did submit designs close to their targets. One possible reason for this contradiction is that, when people compare two different shapes, the difference is not measured in the design (variable) space but in some feature space of the shape, i.e., the perceived difference of a pair can be small when its distance in the design space is large. We can try to verify this reasoning by examining one user's test data (data "Test0" on the website). Fig. 11 compares the target design and samples in the last iteration (iteration 17) which hint that the algorithm understands that the user prefers a roof curve design that has a steep front wind shield and an extended hatchback. In fact, although distance-wise most of these designs are off target, the first five have roof fashions visually close to that of the target while the last one still meets the description but has some variation of its own.

One then surmises that the user focuses mainly on matching the roof style, and the algorithm indeed understands what the user is searching for. In Fig. 12, we show what the user selects in the first four iterations, and from these selections we see a consistent user pursuit and convergence of the search. In the same figure we also show that the design with minimum Euclidean distance does not acquire the roof style the user is looking for and is in fact not picked in the iteration where it appears.

From the above observations, it appears that convergence cannot be shown by the Euclidean distance to the target. Since the user focuses mainly on the roof style design, we formulate a measure that differentiates the roof style as follows. We define style features as seven lengths between key control points of the roof silhouette. The lengths are denoted in Fig. 13 and the measure of difference is defined as the Euclidean distance in the feature space spanned by the seven lengths. With this new measure definition, we examine our earlier conclusion quantitatively. Fig. 14(a) and 14(b) show scatter plots of the sampled designs in a reduced space corresponding to the feature measure and the Euclidean measure accordingly. Each circle represents a sampled design and the square represents the target. The inner radius of a circle indicates when it first comes into existence while the outer radius shows when it is abandoned, in both cases the smaller the radius the earlier the event occurs.

Two observations are made: Firstly, most of the thick circles, representing repeatedly selected designs, are close to the target. This verifies the assumption that the user mainly focuses on matching the roof design; secondly, the feature map has most of the larger circles around the target while smaller ones are away, while in the Euclidean map some of the larger circles are scattered away from the target. The implication is that while no knowledge about the user is posed on the algorithm *a priori*, it "learns" what the user is looking for and generates designs close to the target in the unknown feature space, rather than in the explicit design space.

It should be noted that there are two factors that may affect convergence. Recall that in EGO Search we have a balance between exploration and exploitation. Every new sample is forced to be away from existing ones, and therefore it may not be close to the target in the feature or the Euclidean space. In fact, in the highlighted test data shown above, the final submitted design (the one with Euclidean distance to the target of 1.55, as shown in Fig. 11), appears as early as the fourth iteration but nothing better is found afterwards. One can verify this from Fig. 14(a) where the final submission is shown by the thickest circle close to the target. Therefore, while a balance clearly should be made, how the weights shall be tuned needs more study. Further, the GA implementation in each EGO Search iteration uses a fairly

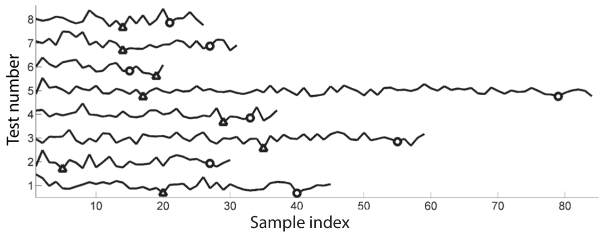


Fig. 9. Normalized euclidean distance from each sampled design to the target in each test. The circled design is the one that submitted by the user, while the triangle design has the lowest euclidean distance to the target.

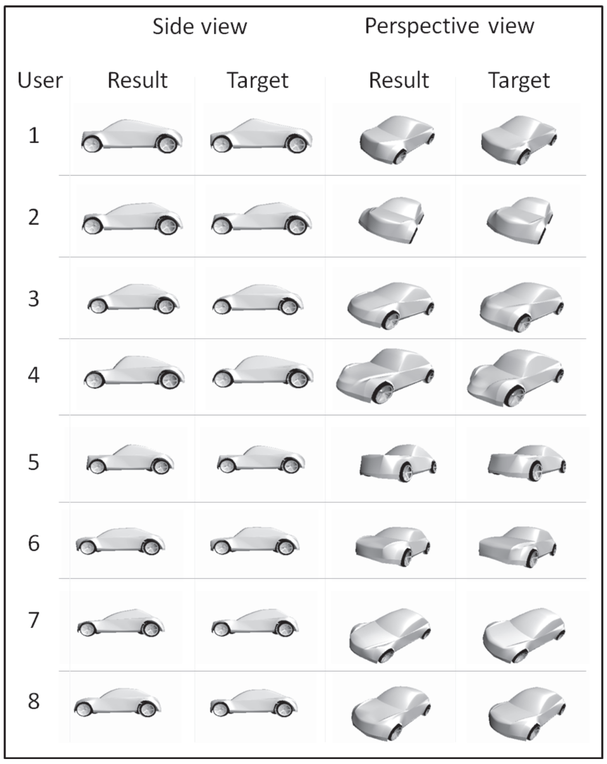


Fig. 10. Visual comparison between user test results and the targets from side and perspective views.

low number of generations with a small population. Therefore, the sampled point may not be the optimum of the merit function and this can lead to an ineffective search.

5.4 Observed Issues in User Interactions

Besides the issues above, there are two other issues we observed from the users in the pilot test. First, even with a target given, the user can still exhibit inconsistent preference during the interaction. For example, some of the users do not realize that they should rotate the designs when evaluating them, though they have been told they can do it. Once they realize in the middle of the interaction process that they can rotate the image, their perception of the shapes changes, and so does their measure of difference between shapes. Second, the specific users appeared rather insensitive to shape

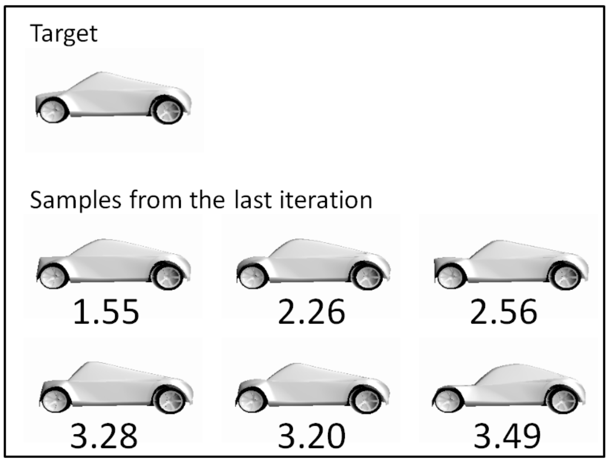


Fig. 11. Visual comparison between samples in the last iteration and the target. Data generated from Test0 on "log.html". Euclidean distances to the target are listed under the designs.

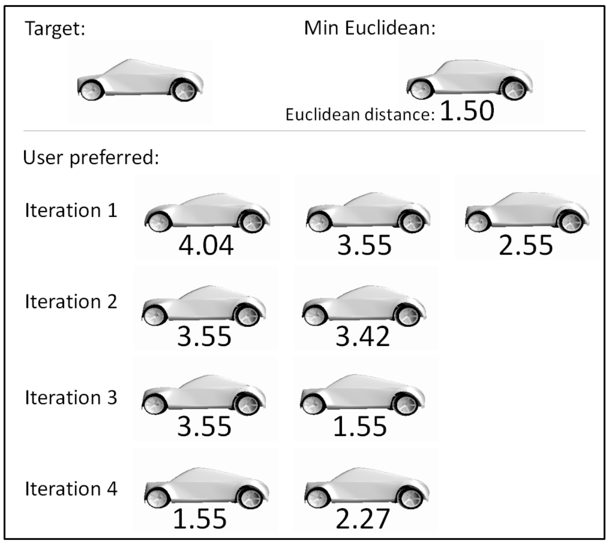


Fig. 12. Designs labeled as preferred in the first four iterations, compared with the target and the one with the minimum Euclidean distance within all samples. Euclidean distances to the target are listed under the designs.

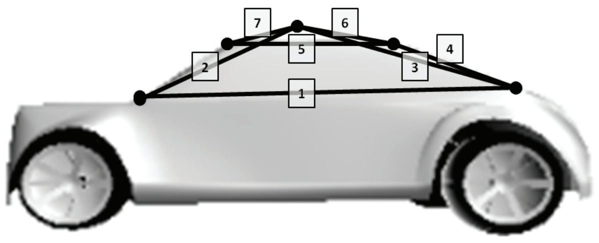


Fig. 13. Features that capture the roof design.

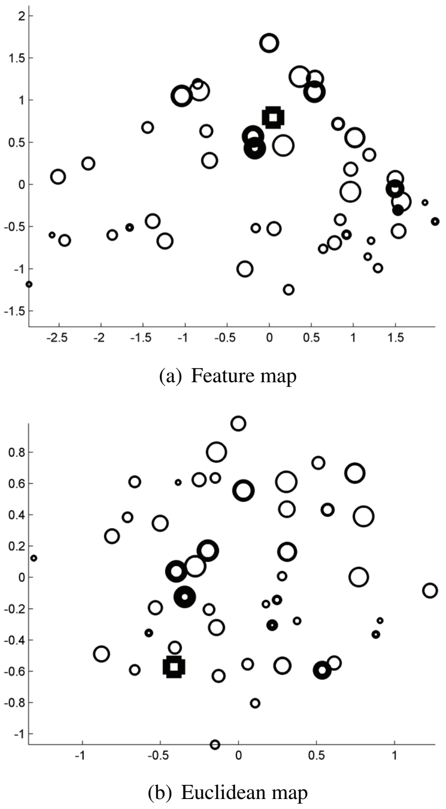


Fig. 14. Samples from data "Test0" in the feature and Euclidean space. Multidimensional scaling is applied to both measures to create 2D visualization of the data.

changes and would miss good designs (following their own measure) during the interaction. Both of these "human mistakes" can confuse the algorithm and make it less effective.

6 Conclusions and Future Work

While incorporating user preference into an engineering optimization framework is appealing, its practical use is still limited due to the difficulty of collecting preference data on qualitative design aspects such as the automotive styling problem. The approach proposed in this paper is a first step towards addressing this issue.

An iterative human-computer interaction was developed: In each iteration, the computer presents a set of designs to the user. The user then assigns binary labels to the designs to imply his or her preference. We showed that this preference elicitation task is akin to a "black-box" optimization problem with binary outputs, and two search strategies were proposed: SVM Search reduces the design space and forces convergence; EGO Search combines exploitation of the observed comparisons and exploration of the unsampled space. The simulated test results showed that both proposed methods are better than GA when only binary feedback is available and that EGO Search outperforms SVM Search when the dimensionality of the design space is larger. The EGO Search algorithm was implemented for a 3D vehicle exterior design application and the pilot tests with real users showed that subjects can successfully locate designs close to their targets.

There are several possibilities for improving and expanding this work. Proper adaptive parameter setting within the merit function in EGO Search is desirable and requires further testing. The computational cost of global optimization of the merit function needs to be reduced further before the proposed algorithm can be used effectively for extensive user interactions. It is also interesting to investigate whether more accurate decision functions can be estimated by using more than two classes. In fact, designs that are switched from preferred to not-preferred should be treated differently from designs that are not preferred from the beginning. The rotation matrices recorded during the interaction can also provide insights on what features the user is sensitive to when she evaluates the difference between a target and the samples. Such insights should be incorporated into the learning process to create more accurate merit functions. Another important direction of future investigation is how to utilize preferences collected from different individuals. Some preliminary tests show that the search can be made more efficient if the algorithm incorporates history from previous users into a current session. Finally, the data collected from an interactive session can be treated as revealed choices from a simulated market, and so combining search process records from many users may provide insights and possibly lead to preference models for groups of users or market segments.

Achnowledgements

This work was partially supported by the Donald C. Graham Chair Endowment and the Department of Mechanical Engineering at the University of Michigan. This support is gratefully acknowledged. We also acknowledge the assistance and test participation provided by members of the Optimal Design Lab at the University of Michigan, and the insights provided by Richard Gonzalez, Professor of Psychology, University of Michigan.

References

[1] Wassenaar, H., Chen, W., Cheng, J., and Sudjianto, A., 2005. "Enhancing discrete choice demand modeling for decision-based design". *ASME Journal of Mechanical Design,* **127**, p. 514.

[2] Kumar, D., Hoyle, C., Chen, W., Wang, N., Gomez-Levi, G., and Koppelman, F., 2007. "Incorporating customer preferences and market trends in vehicle package design". In Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, no. DETC2007-35520.

[3] Li, H., and Azarm, S., 2000. "Product design selection under uncertainty and with competitive advantage". *ASME Journal of Mechanical Design,* **122**(4), pp. 411–418.

[4] Michalek, J., Feinberg, F., and Papalambros, P., 2005. "Linking marketing and engineering product design decisions via analytical target cascading". *Journal of Product Innovation Management,* **22**(1), pp. 42–62.

[5] Kuhfeld, W., Tobias, R., and Garratt, M., 1994. "Efficient experimental design with marketing research applications". *Journal of Marketing Research,* **31**(4), pp. 545–557.

[6] Hoyle, C., et al., 2009. "Optimal experimental design of human appraisals for modeling consumer preferences in engineering design". *ASME Journal of Mechanical Design,* **131**(7).

[7] Toubia, O., Hauser, J., and Simester, D., 2004. "Polyhedral methods for adaptive choice-based conjoint analysis". *Journal of Marketing Research,* **41**(1), pp. 116–131.

[8] Abernethy, J., Evgeniou, T., Toubia, O., and Vert, J., 2007. "Eliciting consumer preferences using robust adaptive choice questionnaires". *IEEE Transactions on Knowledge and Data Engineering*, pp. 145–155.

[9] Netzer, O., Toubia, O., Bradlow, E., Dahan, E., Evgeniou, T., Feinberg, F., Feit, E., Hui, S., Johnson, J., Liechty, J., et al. "Beyond conjoint analysis: Advances in preference measurement". *Marketing Letters,* **19**(3), pp. 337–354.

[10] Takagi, H., et al., 2001. "Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation". *Proceedings of the IEEE,* **89**(9), pp. 1275–1296.

[11] Tokui, N., and Iba, H., 2000. "Music composition with interactive evolutionary computation". In GA2000. Proceedings of the third International Conference on Generative Art.

[12] Johanson, B., and Poli, R., 1998. "Gp-music: An interactive genetic programming system for music generation with automated fitness raters". *Genetic programming*, pp. 181–186.

[13] Kim, H., and Cho, S., 2000. "Application of interactive genetic algorithm to fashion design". *Engineering Applications of Artificial Intelligence,* **13**(6), pp. 635–644.

[14] Kelly, J., and Papalambros, P., 2007. "Use of shape preference information in product design". In International Conference on Engineering Design, Paris, France, Citeseer.

[15] Petiot, J., and Grognet, S., 2006. "Product design: a vectors field-based approach for preference modelling". *Journal of engineering design,* **17**(3), pp. 217–233.

[16] MacDonald, E., Gonzalez, R., and Papalambros, P., 2009. "Preference inconsistency in multidisciplinary design decision making". *ASME Journal of Mechanical Design,* **131**, p. 031009.

[17] Tong, S., and Koller, D., 2002. "Support vector machine active learning with applications to text classification". *The Journal of Machine Learning Research,* **2**, pp. 45–66.

[18] Vapnik, V., 1998. *Statistical learning theory*, Vol. 2. Wiley New York.

[19] Osugi, T., Kim, D., and Scott, S. "Balancing exploration and exploitation: a new algorithm for active machine learning". In Data Mining, Fifth IEEE International Conference on, IEEE, pp. 8–pp.

[20] Baram, Y., El-Yaniv, R., and Luz, K., 2004. "Online choice of active learning algorithms". *The Journal of*

Machine Learning Research, **5**, pp. 255–291.

[21] Basudhar, A., and Missoum, S. "An improved adaptive sampling scheme for the construction of explicit boundaries". *Structural and Multidisciplinary Optimization*, pp. 1–13.

[22] Chang, C., and Lin, C., 2001. LIBSVM: A library for support vector machines.

[23] Jones, D., Schonlau, M., and Welch, W., 1998. "Efficient global optimization of expensive black-box functions". *Journal of Global optimization,* **13**(4), pp. 455–492.

[24] Sacks, J., Welch, W., Mitchell, T., and Wynn, H., 1989. "Design and analysis of computer experiments". *Statistical science,* **4**(4), pp. 409–423.

[25] Jones, D., 2001. "The DIRECT global optimization algorithm". *Encyclopedia of Optimization,* **1**, pp. 431–440.

[26] Chipperfield, A., and Fleming, P., 1995. "The MATLAB genetic algorithm toolbox". In Colloquium Digest-IEE, Citeseer, pp. 10–10.