Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2011
August 28-31, 2011, Washington D.C., United States

DETC2011-48316

DESIGN PREFERENCE ELICITATION USING EFFICIENT GLOBAL OPTIMIZATION

Yi Ren*
Doctoral Candidate, Optimal Design Laboratory
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan, 48104
Email: yiren@umich.edu

Panos Y. Papalambros
Professor, Optimal Design Laboratory
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan, 48104
Email: pyp@umich.edu

ABSTRACT

We seek to elicit individual design preferences through user-computer interaction. During an iteration of the interactive session, the computer presents a set of designs to the user who then picks any preferred designs from the set. The computer learns from this feedback and creates the next set of designs using its accumulated knowledge to minimize a merit function. Under the hypothesis that user responses are deterministic, we show that an effective query scheme is akin to the Efficient Global Optimization (EGO) algorithm. Using simulated interactions, we discuss how the merit function form and user preference sensitivity can affect search efficiency and hence the time to complete an interactive session. We demonstrate the proposed algorithm in the design of vehicle exteriors.

NOMENCLATURE

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

1 INTRODUCTION

This 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. Researches 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]) have demonstrated how these different disciplines can be integrated. Most of these demonstrations utilize preference models with only a small number of variables to make optimization manageable. While Kumar et al. [2] showed that overall exterior appearance of a vehicle has a great effect on consumer

*Address all correspondence to this author.

purchase decisions, capturing preferences for a complex three-dimensional (3D) parameterized vehicle shape requires a large number of design variables.

Moreover, capturing the right design variables for an essentially qualitative or holistic design problem such as vehicle styling is difficult to accomplish using market (revealed) data or user survey (stated) data preferences. Researchers have recognized this problem (e.g., Hoyle et al. (2009) on D-efficient sampling [5], Abernethy et al. (2007) on adaptive sampling [6]) and proposed techniques for sampling data efficiently to improve estimation of an underlying model. An alternative is to use a human-computer interaction environment and solicit preferences directly from individuals, for example, using interactive evolutionary computation (IEC). Vectorized (in genetic algorithms) and structured (in genetic programming) design representations have been utilized in IEC to elicit user preferences on music [7] [8], dressing [9], vehicle designs [10] and many others. A comprehensive review in this field can be found in Takagi [11]. Researchers in IEC have realized that user fatigue will occur during the process of assigning fitness and will cause deterioration of user responses. Remedies proposed include reduced fitness scale, e.g., seven rating levels as oppose to a hundred. Another issue in using IEC is that ratings are relative, i.e., rating levels from different iterations are not comparable. With these issues in mind, we explore here the question of what convergence behavior can we achieve with an algorithm that uses only binary comparisons in each iteration, and allows comparisons between new and previously preferred designs. In this paper, we introduce a statistical learning algorithm to collect individual preferences interactively using only binary comparison data, without the cumbersomeness of fitness assigning or interpretation, and we apply it to vehicle exterior style designs. We demonstrate that this interaction can successfully elicit user preferences and help users to search for their most preferred designs, ensuring an appealing and meaningful user experience.

We formulate the interaction problem as follows. Let $\mathscr{D} \subset \mathbb{R}^p$ be the design space where any $\mathbf{x} \in \mathscr{D}$ 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 $f(\mathbf{x}_1, \{\mathbf{x}_i\}_{i=2}^l) : (\mathscr{D}, \mathscr{D}^{l-1}) \to [0,1]$ 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 $\mathscr{D}$ defined as follows.

Definition 1. *A design $\mathbf{x}_1$ belongs to the most preferred region $\mathscr{D}_+ \subseteq \mathscr{D}$ if and only if $\forall \{\mathbf{x}_i\}_{i=2}^l \in \mathscr{D}^{l-1}$ and $\forall l = 2, 3, \ldots$, 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. Mathematically, this means that the response function can be modeled as an indicator function defined on $\mathscr{D}$ and is independent of the parameter set $\{\mathbf{x}_i\}_{i=2}^l$, as given in Definition 2:

Definition 2. *User response as an indicator function:*

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

The definition implies that each design within $\mathscr{D}$ is associated with a definite label and is independent of other accompanying designs. By Definition 1, $\mathscr{A} = \mathscr{D}_+$. Thus learning user preference can be treated as a classification problem to approximate $\mathscr{D}_+$ by querying labels of sample points in $\mathscr{D}$. Ren and Papalambros [12] showed that an active learning technique [13] can be directly adopted to tackle this problem.

However, this model prevents users from selecting relatively better designs in the presented set when none of them is close to his or her preference. The second type of question thus comes into play: "Which designs in the set do you prefer more?". By introducing an arbitrary utility function $u(\mathbf{x}) : \mathscr{D} \to \mathbb{R}$, 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. With these, the response function for this type of question can be modeled as follows.

Definition 3. *Response with utility and clustering:*

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

Notice that when the utility function is continuous and bounded on $\mathscr{D}$ 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 $\mathscr{D}_+$ only contains $arg\max_{\mathbf{x}}\ u(\mathbf{x})$. Indeed, if there exists $\mathbf{x} \in \mathscr{D}_+$ 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 $\mathbf{x} \notin \mathscr{D}_+$. With these modeling assumptions and with this question, the problem is a "black-box" optimization one with binary outputs. Ren and Papalambros [14] proposed a Support Vector Machine (SVM) search algorithm for this problem that creates the next set of designs based on previously established decision boundaries. For this special binary output situation, the algorithm was shown to be more effective than the DIRECT [15] and genetic algorithm. In this paper, we show that the use of efficient global optimization (EGO) [16] can further enhance the search.

In the proposed algorithm we use two main heuristics: (1) The samples to query should have high predicted utility values, and (2) the samples should be far from any existing sample for exploration purposes. Implementation of these heuristics involves two steps: (a) Modeling and optimizing a utility function on $\mathscr{D}$ based on the binary feedback, and (b) modeling and optimizing a minimum distance function to facilitate exploration. As we will review in Section 2, the same heuristic has been used in EGO where the responses are predicted using a kriging model and the minimum distance function is substituted by a mean square error function. We demonstrate in Section 3 that the proposed implementation has better performance than the previously proposed SVM search algorithm on all test functions. Due to the heuristic nature of the algorithm, we explore the impact of parameters used in the algorithm in Section 4. In Section 5 a demonstration with a 3D vehicle styling problem is presented, and we conclude with some observations on limitations in Section 6.

2 Background

This section reviews EGO for global optimization and SVM for classification. EGO utilizes a kriging (linear mixed) model in each iteration and attempts to find the global optimal solution by refining this kriging model through the iteration process. SVM is used here for classification purposes since only binary outputs are available. Two important observations are made: (1) although the derivation of kriging estimators is different from that of the estimators for an SVM decision function, they both predict the response as a weighted sum of kernel distances from the sample of interest to all evaluated samples; (2) the mean square error function used in EGO can be considered as a minimum distance function that is differentiable everywhere, and using this smooth function enhances the performance of the search algorithm.

2.1 Kriging

Denote $\{\mathbf{x}_i\}_{i=1}^l$ as a set of l observations and $\{y_i\}_{i=1}^l$ as their responses. The generalized kriging model with k parameters $\beta_h,\ h=1,...,k$ has the following form:

$$y(\mathbf{x}_i) = \sum_{h=1}^{k} \beta_h f_h(\mathbf{x}_i) + \varepsilon(\mathbf{x}_i),\ i=1,...,l. \tag{3}$$

Here $\{f_h(\mathbf{x})\}_{h=1}^k$ is a set of linear or nonlinear functions of $\mathbf{x}$, and $\varepsilon(\mathbf{x})$ is a random error. The correlation between $\varepsilon(\mathbf{x}_i)$ and $\varepsilon(\mathbf{x}_j)$ is assumed to be:

$$\begin{aligned}&\mathrm{Corr}[\varepsilon(\mathbf{x}_i),\varepsilon(\mathbf{x}_j)] = \exp\left(-\lambda^{\text{kriging}}||\mathbf{x}_i-\mathbf{x}_j||_2^2\right),\\ &\lambda^{\text{kriging}} \geq 0,\end{aligned} \tag{4}$$

where λ^{kriging} is the spread parameter that governs a distance between two samples. It is usually practical to simplify the linear model term $\sum_{h=1}^k \beta_h f_h(\mathbf{x}_i)$ as a constant β, in which case we have the ordinary kriging model:

$$y(\mathbf{x}_i) = \beta + \varepsilon(\mathbf{x}_i),\ i=1,...,l. \tag{5}$$

The kriging estimate $\hat{\beta}$ is derived using a maximum likelihood estimation and thus the prediction $\hat{y}$ can be evaluated by Equation (6), see Sacks [17] for a derivation.

$$\begin{aligned}\hat{y}(x) &= \hat{\beta} + \mathbf{r}^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\hat{\beta}),\\ \text{where:}&\\ r_i(x) &= \mathrm{Corr}[\varepsilon(\mathbf{x}_i),\varepsilon(\mathbf{x})]\\ &= \exp\left(-\lambda^{\text{kriging}}||\mathbf{x}_i-\mathbf{x}||_2^2\right),\ \forall i=1,...,l,\\ R_{ij} &= \mathrm{Corr}[\varepsilon(\mathbf{x}_i),\varepsilon(\mathbf{x}_j)]\\ &= \exp\left(-\lambda^{\text{kriging}}||\mathbf{x}_i-\mathbf{x}_j||_2^2\right),\ \forall i,j=1,...,l,\\ \hat{\beta} &= \frac{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}.\end{aligned} \tag{6}$$

2.2 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 $\{y_i\}_{i=1}^l \in \{-1,1\}^l$. Notice that in the numerical setup of an SVM problem, we use binary labels $\{-1,1\}$ rather than $\{0,1\}$.

$$\begin{aligned}\min_{\mathbf{w},b,\xi}\quad & \mathbf{w}^T\mathbf{w} + C\sum_{i}^{l}\xi_i,\\ s.t.\quad & y_i(\mathbf{w}^T\mathbf{x_i}+b) \geq 1-\xi_i,\\ & \xi_i \geq 0,\ i=1,...,l.\end{aligned} \tag{7}$$

The solution to problem (7) finds a linear decision function $\hat{y} = \mathbf{w}^T\mathbf{x} + b$ that minimizes a merit function combining 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 Problem (7) 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 $\psi(\cdot): \mathscr{D}\rightarrow\mathscr{F}$ represents any mapping of $\mathbf{x}$ to an unknown feature space $\mathscr{F}$. Such a method is referred to as a kernel trick. The solution for Problem (7), with a Gaussian kernel $K_{ij}=\exp(-\lambda^{\text{SVM}}||\mathbf{x}_i-\mathbf{x}_j||_2^2)$, can be summarized as:

$$\hat{y}(\mathbf{x})=\sum_{i=1}^{l} y_i\alpha_i\exp(-\lambda^{\text{SVM}}||\mathbf{x}-\mathbf{x}_i||_2^2)+b, \tag{8}$$

where α_i and b are solutions to the dual problem of problem (7). One can see that the solution to a soft-margin SVM, Equation (8), and that used in the kriging model, Equation (6), have the same weighted (kernel) distance form. Indeed, the SVM decision function $\hat{y}$ can be considered as a prediction model of utilities, with current support vectors (samples with $\alpha_i>0$) having unit utilities, i.e., 1 for preferred designs and -1 for not-preferred ones.

2.3 Mean Square Error and Minimum Distance

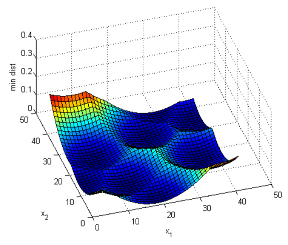
In a kriging model, the mean square error s^2 is a measure for how much uncertainty we have in a prediction from the model. Intuitively, the uncertainty is high when the sample to be predicted is far away from any existing samples and, as a special case, the uncertainty becomes zero at any existing sample. We directly provide this error under the ordinary kriging model in Equation (9), see Sacks [17] for a derivation.

$$s^2(\mathbf{x})=[1-\mathbf{r}^T\mathbf{R}^{-1}\mathbf{r}+\frac{(1-\mathbf{1}^T\mathbf{R}^{-1}\mathbf{r})^2}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}] \tag{9}$$

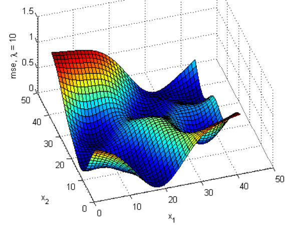
A minimum Euclidean distance function d^2 defined on $\mathscr{D}$ with a set of l existing samples $\{\mathbf{x}_i\}_{i=1}^l$, on the other hand, is defined as:

$$d^2(\mathbf{x})=\min_i\{||\mathbf{x}-\mathbf{x}_i||_2^2\}_{i=1}^l. \tag{10}$$

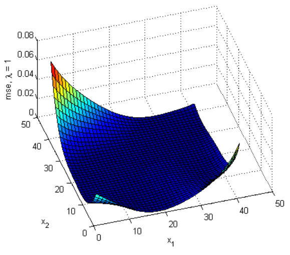
Figures 1(a) and 1(b) illustrate the similarity between the minimum distance function, Equation (10), and the mean square error function, Equation (9), with the correlation parameter $\lambda^{\text{kriging}}=10$ on a given sample set. From Figures 1(c) and 1(d), we observe that for the mean square error function, increasing its spread λ^{kriging} lifts the entire $\mathscr{D}$ to be equally in favor except for points close to existing samples, while decreasing its spread shifts the preference to regions that are far away from all existing samples. Therefore, λ^{kriging} controls how much exploration is required for a search.



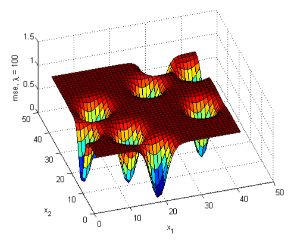
(a) Minimum distance function



(b) Mean square error function, $\lambda^{\text{kriging}}=10$



(c) Mean square error function, $\lambda^{\text{kriging}}=1$



(d) Mean square error function, $\lambda^{\text{kriging}}=100$

FIGURE 1. Comparison between the minimum distance function and the mean square error function with different Gaussian spread λ^{kriging} on a fixed sample set.

3 Proposed Algorithm

A search algorithm for user-computer interaction with binary choice feedback can be built in the same fashion as 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. Algorithm 1 provides a summary of the steps.

Several details need to be clarified: (1) The merit function we use in this section takes the form of $f(x)=-(w_1\hat{y}+w_2s^2)$, where $\hat{y}$ is the predicted utility and s^2 is the mean square error.

Algorithm 1: Proposed Search Algorithm

```
input  : Design space $\mathscr{D}$, 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 ($\mathscr{D}$,$l$);

// 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
        $\mathbf{x}_j$ = Scatter ($g(\mathbf{x})$,$\mathbf{X}$,$\mathscr{D}$,$l$); 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
```

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 (i) make the two function values of the same scale and (ii) start the search with an emphasis on exploration and 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 10^6 since we consider all user input to be correct and little training error is allowed. This assumption, however, does not always hold in real user interactions; (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 $\mathscr{D}$ with high upper bounds for both $\hat{y}$ and s^2 [16]. To this end, it is not clear whether this method will be effective when the dimensionality of $\mathscr{D}$ is high. As an alternative, the Matlab GA toolbox [?] is used to optimize the merit function in the present study.

Figure 2 shows comparisons on standard optimization test functions between the proposed algorithm and that in [14]. Readers may refer to the Appendix for all function forms. For each algorithm and each function, ten tests are executed and the error is calculated as the current minimum gap from the theoretical optimum. We use this form of error because in real user interaction, it is always possible to ask the question: "Among all the preferred designs, which ones are the most preferred?". The solid line and bars show the mean error and its standard deviation of the proposed algorithm, respectively, while the dotted ones show those from the SVM search algorithm. One can see that, overall the proposed algorithm has more reliable performance than SVM search. The most notable observation is that while the SVM search algorithm has significantly decreasing capability in searching in high dimensional space, the proposed algorithm provides an encouraging performance. This would be an advantage of the proposed algorithm when the dimensionality of the design space is high in real user applications.

4 Discussion of the Proposed Algorithm

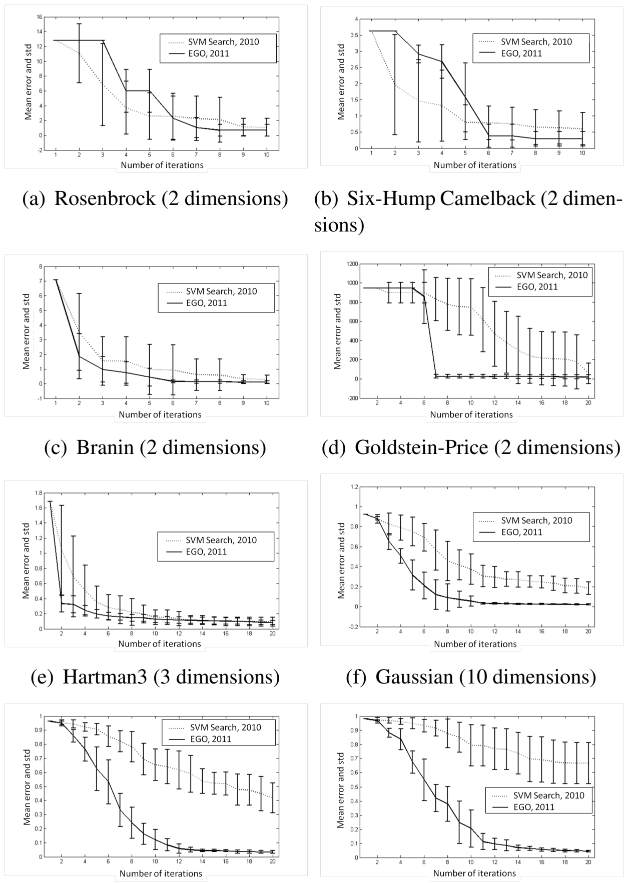
In this section we investigate a few performance questions related to the proposed algorithm: (1) How do we set algorithm parameters? (2) How many dimensions can the algorithm handle effectively, since this would determine the maximum number of variables we can use in a user-computer interaction? (3) How does user preference sensitivity affect the search?

4.1 Parameters used in the merit function

We investigate the influence of the form of the merit function on the search result. Two types of merit function are of interest. One form of merit function, as we used in the previous section, is a weighted sum of the utility and the mean square error: $w_1\hat{y} + w_2 s^2$. The other form, introduced by Jones et al. [16], evaluates the expected improvement on $\mathscr{D}$ and is rooted back to its kriging model and the mean square error function. In a maximization problem, given f_{max} as the best point so far, this expected improvement has the following form:

$$E[\max(Y - f_{max}, 0)] = (\hat{y} - f_{max})\Phi\left(\frac{\hat{y} - f_{max}}{s}\right) + s\phi\left(\frac{\hat{y} - f_{max}}{s}\right). \tag{11}$$

Here on the left hand side, the improvement is expressed as the expected amount that the random variable Y on $\mathscr{D}$ can exceed the present best value f_{max}. Notice that in this study, since we set the weight C in SVM at a large value, f_{max} will always be very close to 1. By modeling the random variable Y to be normally



(a) Rosenbrock (2 dimensions) (b) Six-Hump Camelback (2 dimensions) (c) Branin (2 dimensions) (d) Goldstein-Price (2 dimensions) (e) Hartman3 (3 dimensions) (f) Gaussian (10 dimensions) (g) Gaussian (13 dimensions) (h) Gaussian (15 dimensions)

FIGURE 2. Comparison between SVM Search [14] and the proposed algorithm.

distributed as $N(\hat{y}, s^2)$, we derive the right hand side formula with $\Phi(\cdot)$, $\phi(\cdot)$ being the cumulative distribution function (CDF) and probability distribution function (PDF) for a standard normal distribution, respectively.

Below we investigate the influence of parameters of these two merit functions. For the weighted sum function, we test a set of weights: $w_1 = 1, w_2 = (t/i)^k - 1,\ k = 1,2,3$, where t is the maximum number of iterations and i is the current iteration number. The weights are set this way to examine how much exploration is needed during the search and diminish the exploration emphasis towards the end of the process. For the expected improvement merit function, we are interested in the Gaussian spread λ used in evaluating the mean square error function. We test a set $\lambda = 10/i, 100/i, 1000/i, 10000/i$ to see what scale of λ should be recommended. It should be mentioned that the similarity matrix R will be ill-conditioned when the spread λ is small. This is because all values off the diagonal in $\mathbf{R}$ will be closer to 1 with a smaller λ while the diagonal elements are always 1's. Thus the minimum λ during the process is empirically set to 1

TABLE 1. Impact of the weights of the weighted-sum merit function

error $\mu(\sigma)$	$k = 1$	2	3
Branin	0.140 (0.147)	0.233 (0.333)	0.427 (0.468)
Camel	0.578 (0.298)	0.588 (0.329)	0.584 (0.424)
Hartman3	0.167 (0.053)	0.173 (0.038)	0.149 (0.035)
Rosenbrock	2.582 (1.314)	1.175 (1.282)	3.777 (2.712)

TABLE 2. Impact of the spread of the expected improvement merit function

error $\mu(\sigma)$	$\lambda^{kriging} = 10/i$	$100/i$	$1000/i$
Branin	0.300 (0.122)	0.126 (0.102)	0.051 (0.039)
Camel	1.057 (0.554)	0.559 (0.356)	1.356 (0.704)
Hartman3	0.208 (0.065)	0.232 (0.038)	0.226 (0.067)
Rosenbrock	5.844 (5.686)	0.642 (0.135)	1.074 (0.771)

(when the maximum iteration number is set to 10). Again, λ decreases along the iteration to shift the priority from exploration to exploitation.

Tables 1 and 2 show simulation results from various test functions using the proposed merit functions. We run ten tests for each combination of function and spread; each test contains ten iterations; within each iteration eight samples are labelled. The reported numbers are the mean errors after the last iteration and their standard deviations. From the results it can be concluded that although some parameter values achieve better and more stable performance over others, there is no universally recommended parameter values for the merit functions at this point. Also, the sensitivity of the resulting error with regard to the parameters is different for different test functions. Therefore the use of adaptive parameters should be further investigated.

4.2 Dimensionality and user sensitivity

It is intuitively true that with a design of higher dimensionality, one expects this search to be less effective. On the user side, if the utility function of a user has an exact optimal solution while it is flat elsewhere, finding this solution will be hard. We are interested in how the proposed algorithm will work for different dimensions and utilities. As a preliminary test, we use 10D, 15D and 20D Gaussian functions as the utilities and changes in their spread λ^{utility} in the set of $[0.5, 1, 2, 4]$. We run ten tests for each combination of dimensionality and spread; each test contains twenty iterations; within each iteration eight samples are labelled. The merit function used here is $\hat{y} + (t/i - 1)s^2$, where t is the maximum iteration number and i is the current iteration number. Table 3 shows the mean error (μ) that the algorithm

TABLE 3. Impact of dimensionality and spread of the utility of the proposed algorithm

error $\mu(\sigma)$	dim = 10	15	20
$\lambda^{\text{utility}} = 0.5$	0.026 (0.005)	0.041 (0.007)	0.066 (0.018)
1	0.046 (0.016)	0.074 (0.010)	0.221 (0.139)
2	0.068 (0.026)	0.149 (0.017)	0.305 (0.094)
4	0.147 (0.039)	0.265 (0.036)	0.481 (0.125)

reaches at the 20th iteration and its standard deviation (σ).

Although simulations are not direct reflection of how the algorithm will perform in real situations, there are a few observations we can derive from these results: (1) Noticing that the Gaussian function ranges from 0 to 1, the algorithm achieves an error less than 10% only when the spread is large (low sensitivity of user preference) or the dimensionality is low; the algorithm has very limited capability when dimensionality increases beyond twenty; (2) the algorithm has fairly consistent performance since the standard deviation of the error at the 20th iteration is low.

4.3 Computational cost

For the proposed algorithm to work effectively, a solution close to the global optimum must be found in each optimization of the merit function. Since the merit function is non-convex, finding such a solution is difficult. Although the Matlab GA toolbox provides a satisfactory solution for the tested problem dimensions, the computational cost of such an implementation is less than acceptable for real human-computer interactions. In fact, when running on an i5-460M CPU with 4G RAM computer, the maximum CPU time spent in one iteration for the 10D Gaussian test function can be up to 200 seconds on average. This value goes up to 350 seconds and 400 seconds on 15D and 20D Gaussian test functions respectively.

5 User Test

This section demonstrates the implementation of the proposed algorithm in a web-based human-computer interaction. The purpose of this interaction is to elicit users' preferences on vehicle exterior styling designs. The demonstration shows that one can fairly efficiently find a target design using the proposed algorithm.

5.1 Software development

To enable real-time online human-computer interactions on designing vehicle styles, a parametric 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, the website is accessible at "http://yirenumich.appspot.com" and Google Chrome 10 is running this application.

5.2 Demonstration

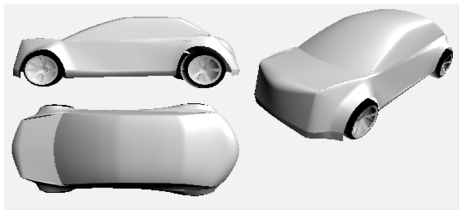
The interaction works as follows: The user accesses the website and sees eight sample designs. The user can then drag and zoom the designs individually to evaluate them visually. Double clicking a design will highlight it and thus label it as a preferred one. Another double click cancels the selection. Once done with the selection, the user can click a "train" button to get the next iteration of eight designs.

To illustrate, we take a random design shown in Figure 3(a) as the target. Figure 5 lists the seven iterations the user goes through to find a design that resembles the target. In the first iteration a random Latin Hypercube design set is created. Since none of the designs is close to the target, the user picks the three designs that have features relatively closer to the target. The second generation has no design resembling the target silhouette. However, the user recognizes a few hood designs that are fairly close to the target hood. Thus these designs are chosen; the one design that has a closer silhouette to the target is chosen from the third iteration and it spawns a few even better ones in the fourth iteration. The user keeps on selecting designs that maintain the hood feature and improve the silhouette until most of the designs are close to the target in the 7th iteration. The final choice is shown in Figure 3(b) and one may see its close resemblance to the target. It should be noted that with respect to the Euclidean distance from the target to the final search result, the two are not as close as a user may feel. Indeed, we plot the Euclidean distances from all sampled designs to the target in Figure 4 to show several interesting observations: The algorithm works in the sense that it reduces the mean and standard deviation of the distances. However, one may notice that during the first iteration, the user missed the relatively best design and chose the worst design with respect to the distance. This shows that a user's utility is not always a monotonic function of the distance, e.g., a Gaussian function, and the proposed algorithm can perform well on arbitrary utility functions as shown in the simulation.

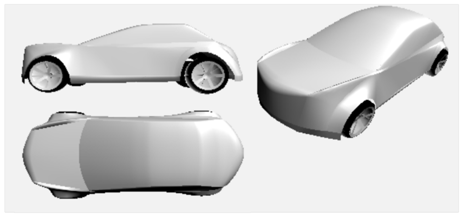
6 Conclusions and Future Work

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

An iterative human-computer interaction was developed: In



(a) Top, side and perspective views of the target.



(b) The design resulting from the search.

FIGURE 3. A comparison between the search result and the target.

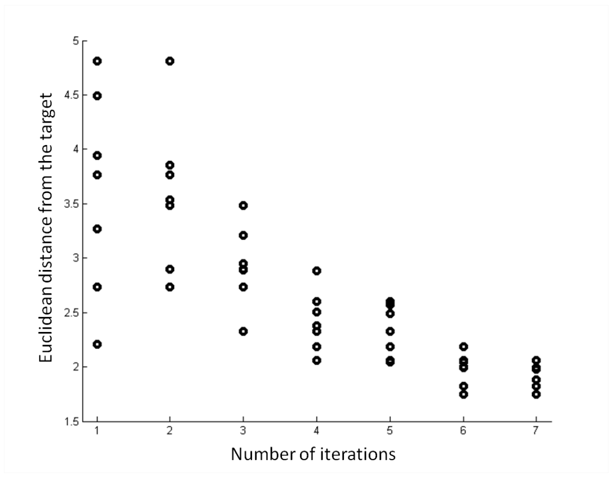


FIGURE 4. Euclidean distance to the target from all sampled designs.

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 we proposed a search strategy inspired by the efficient global optimization algorithm. We showed in a 3D vehicle exterior design demonstration that the algorithm can elicit users' target design fairly effectively. However, as noted in MacDonald et al. [19], the assumption that a user has indeed a target design in mind is questionable. Therefore, examining how such a process will work if this assumption is not satisfied is a subject for further investigation.

There are several possibilities for improving and expanding this work. Proper adaptive parameter setting within the merit function is desirable and would require further testing. The computational cost of a global optimization search on the merit function needs to be reduced further before the proposed algorithm can be used effectively for 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. 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 learning session. Further, while the data collected from an interactive session can be treated as revealed choices from a simulated market, combining search process records from many users may provide insights and possibly lead to preference models for groups of users or market segments.

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". *Journal of Mechanical Design,* ***122***(4), pp. 411–418.

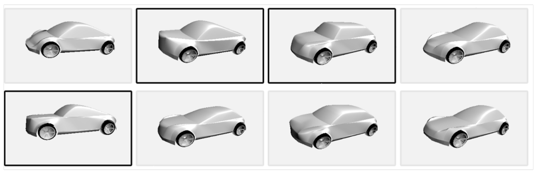
[4] Michalek, J., Feinberg, F., and Papalambros, P., 2005. "Linking Marketing and Engineering Product Design Decisions via Analytical Target Cascading*". *Journal of Product Innovation Management,* ***22***(1), pp. 42–62.

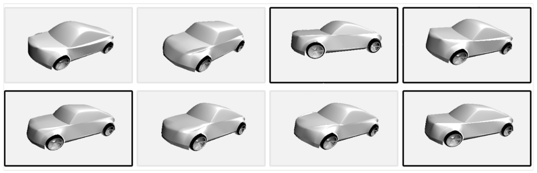
[5] Hoyle, C., et al., 2009. "Optimal experimental design of human appraisals for modeling consumer preferences in engineering design". *ASME Journal of Mechanical Design,* ***131***(7).

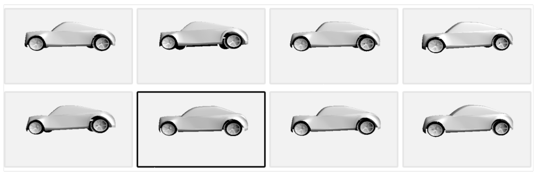
[6] Abernethy, J., Evgeniou, T., Toubia, O., and Vert, J., 2007. "Eliciting consumer preferences using robust adaptive choice questionnaires". *IEEE Transactions on Knowledge and Data Engineering*, pp. 145–155.

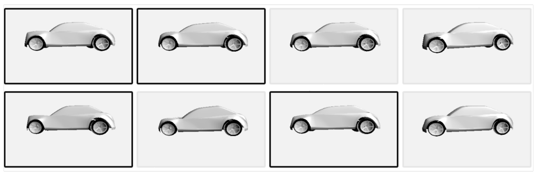
[7] Tokui, N., and Iba, H., 2000. "Music composition with interactive evolutionary computation". In GA2000. Proceedings of the third International Conference on Generative Art.

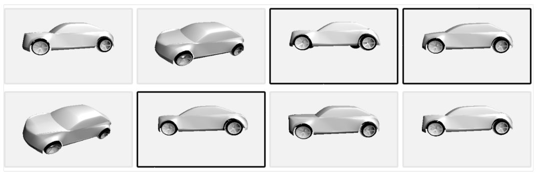
[8] Johanson, B., and Poli, R., 1998. "Gp-music: An interactive genetic programming system for music generation with automated fitness raters". *Genetic programming*, pp. 181–186.

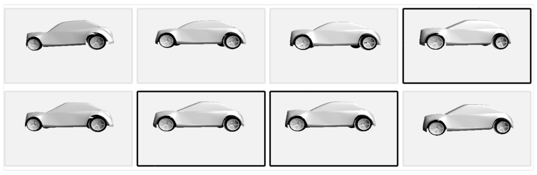

(a) Iteration 1


(b) Iteration 2


(c) Iteration 3


(d) Iteration 4


(e) Iteration 5


(f) Iteration 6

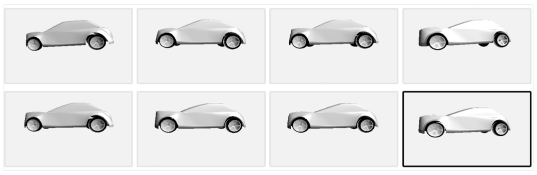

(g) Iteration 7

FIGURE 5. A user-computer interaction example using the proposed search algorithm.

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

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

[11] 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.

[12] Ren, Y., and Papalambros, P., 2011. "Design Preference Elicitation: Exploration and Learning". In Proceedings of the International Conference on Engineering Design, no. 200.

[13] 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.

[14] Ren, Y., and Papalambros, P., 2010. "Design Preference Elicitation, Derivative Free Optimization and Support Vector Machine Search". In Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, no. DETC2010-28475.

[15] Lewis, R., Torczon, V., and Trosset, M., 2000. "Direct search methods: then and now". *Journal of Computational and Applied Mathematics,* ***124***(1-2), pp. 191–207.

[16] 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.

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

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

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

APPENDIX: TEST FUNCTIONS

2D Rosenbrock function:

$$
\begin{aligned}
f(x,y) &= -\left((1-x)^2 + 100(y-x^2)^2\right), \\
\mathscr{D} &= \{(x,y),\ x \in [-1.5, 1.5],\ y \in [-0.5, 1.5]\}.
\end{aligned}
\tag{12}
$$

2D Six-hump Camelback function:

$$f(x,y) = -\left((4-2.1x^2+\frac{x^4}{3})x^2+xy+(-4+4y^2)y^2\right),$$
$$\mathscr{D} = \{(x,y),\ x\in[-3,2],\ y\in[-3,2]\}. \quad (13)$$

2D Branin function:

$$f(x,y) = -\left((y-\frac{5.1x^2}{4\pi^2}+\frac{5x}{\pi}-6)^2+10(1-\frac{1}{8\pi})\cos(x)+10\right),$$
$$\mathscr{D} = \{(x,y),\ x\in[-5,10],\ y\in[-5,10]\}. \quad (14)$$

2D Goldstein-Price function:

$$\begin{aligned}
f(x,y) &= -\left(1.0+a(x,y)^2b(x,y)\right)\left(30.0+c(x,y)^2d(x,y)\right),\\
\textit{where}:&\\
a &= x+y+1.0,\\
b &= 19.0+14.0x+3.0x^2-14.0y+6.0xy+3.0y^2,\\
c &= 2.0x-3.0y,\\
d &= 18.0-32.0x+12.0x^2+48.0y-36.0xy+27y^2,\\
\mathscr{D} &= \{(x,y),\ x\in[-6,6],\ y\in[-5,5]\}. \quad (15)
\end{aligned}$$

3D Hartman3 function:

$$\begin{aligned}
f(\mathbf{x}) &= \sum_{i=1}^{4}\alpha_i\exp\left(-\sum_{j=1}^{3}A_{ij}(x_j-P_{ij})^3\right),\\
\textit{where}:&\\
\alpha &= [1,1.2,1,3.2],\\
A &= \begin{bmatrix}3.0 & 10 & 30\\ 0.1 & 10 & 35\\ 3.0 & 10 & 30\\ 0.1 & 10 & 35\end{bmatrix}, P = \begin{bmatrix}0.3689 & 0.1170 & 0.2673\\ 0.4699 & 0.4387 & 0.7470\\ 0.1091 & 0.8732 & 0.5547\\ 0.0382 & 0.5743 & 0.8828\end{bmatrix},\\
\mathscr{D} &= \{\mathbf{x}\in\mathbb{R}^3,\ x_i\in[0,1]\ \forall i=1,2,3\}. \quad (16)
\end{aligned}$$

*N*D Gaussian function:

$$f(\mathbf{x}) = \exp(-\lambda\sum_{i=1}^{N}(x_i-0.9)^2),$$
$$\mathscr{D} = \{\mathbf{x}\in\mathbb{R}^N,\ x_i\in[-1,1]\ \forall i=1,...,N\}. \quad (17)$$