**Proceedings of the ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference**
**IDETC/CIE 2010**
**August 15-18, 2010, Montreal, Quebec, Canada**

**DETC2010-28475**

# DESIGN PREFERENCE ELICITATION, DERIVATIVE-FREE OPTIMIZATION AND SUPPORT VECTOR MACHINE SEARCH

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

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

## ABSTRACT

*In design preference elicitation, we seek to find individuals' design preferences, usually through an interactive process that would need only a very small number of interactions. Such a process is akin to an optimization algorithm that operates with point values of an unknown function and converges in a small number of iterations. In this paper, we assume the existence of individual preference functions and show that the elicitation task can be translated into a derivative-free optimization (DFO) problem. Different from commonly-studied DFO formulations, we restrict the outputs to binary classes discriminating sample points with higher function values from those with lower values, to capture people's natural way of expressing preferences through comparisons. To this end, we propose a heuristic search algorithm using support vector machines (SVM) that can locate near-optimal solutions with a limited number of iterations and a small sampling size. Early experiments with test functions show reliable performance when the function is not noisy. Further, SVM search appears promising in design preference elicitation when the dimensionality of the design variable domain is relatively high.*

## NOMENCLATURE

| | |
|---|---|
| $\mathscr{D}$ | design space with dimensionality denoted by $n$. |
| $\mathscr{F}$ | feature space for classification purpose. |
| $k$ | maximum number of iterations. |
| $l$ | maximum sampling size in each iteration. |
| $\{\mathbf{x}_i\}_l$ | a sample set on $\mathscr{D}$ containing $\mathbf{x}_i$ for $i = 1,...,l$. |
| $\{y_i\}_l$ | labels on $\{\mathbf{x}_i\}_l$. |

[*]Address all correspondence to this author.

## INTRODUCTION

Designers care about user preferences and have long been attempting to quantify and capture them. Methods like conjoint analysis [1] and Kansei engineering [2] have been applied successfully in various product areas [3] [4]. Knowledge elicitation is generally defined as the process of capturing human knowledge for computer program use, and it has been extensively studied in artificial intelligence and human-computer interaction (HCI) [5]. Design elicitation is also widely studied in HCI research as a tool for better interface designs [6] [7]. Preference or utility elicitation is widely studied also in decision analysis and computer science [8]. Here we coin the term *design preference elicitation* as the task of eliciting a person's preferences regarding an attribute or other aspect of an artifact's design, where the person can be a user, designer, manager, or anyone with the requisite standing. Design preference elicitation should be efficient, i.e., operate preferably in real-time and without tiring the user, and accurate, i.e., effective for sufficiently complex designs. Developments of interactive evolutionary computing [9] [10] offer a good promise, but in practice they lack convergence properties due to human fatigue especially when the design variable space has high dimensionality [9].

Copyright © 2010 by ASME

In this paper, we consider an interactive process with discrete iterations. In each stage, the computer generates sample designs using its own knowledge about the user. This knowledge is acquired and updated from user feedback, namely, answers to preference questions posed by the computer. Various forms of user feedback, e.g., rates [11] [12] and ranking, have been used in studies with similar interactive procedures. In the present study, we assume that a natural, credible design preference elicitation occurs by asking the user to assign binary labels, selecting the preferred designs from the rest. We also assume certain properties exist in the collected data: the user cannot categorize all designs into a single class, i.e., selection is always required in the feedback; second, responses are transitive, meaning that if A is better than B and B is better than C, then A must be better than C; and third, all responses are deterministic.

A further major assumption is that human preferences have functional representations. Formally, let $\mathscr{D}$ be the design space and $f : \mathscr{D} \to \Re$ be an individual preference function defined on $\mathscr{D}$. For any individual, a response is constructed by first taking in a set of $l$ sample points $\{\mathbf{x}_i\}_l$ and then evaluating and clustering the function values $\{f(\mathbf{x}_i)\}_l$ into two classes with labels $\{y_i\}_l$.

In this setting, the following question is to be investigated: Within a fixed number of stages and limited sampling size, how can we get closer to the optimum of $f$ with the binary class outputs $\{\mathbf{x}, y\}$? Realizing that this question resides in the category of DFO problems, or direct search problems as in [13], an appropriate search algorithm would exist for finding the solution. However, among search algorithms in this category, few are designed to handle binary class outputs. The problem we are interested in differs further from traditional DFOs in that we care about short term results rather than long term convergence, since the iterations are human evaluations that should be kept short. Therefore, this paper introduces a potential solution using a support vector machine (SVM) search that adaptively refines and reduces the design space towards a near-optimal region.

The rest of this paper is organized as follows. Section 2 reviews two existing algorithms that have the potential to address the problem at hand. An introduction of SVM then follows to provide a basis for the discussion in Section 3 where we describe in detail the proposed search algorithm. Section 4 compares the performance of the three algorithms in simulations. We also give a demonstration of how the proposed algorithm can be used in real-time user interaction environment. Section 5 concludes the study.

## RELATED WORK

We discuss two methods from the literature, namely genetic algorithms (GA) and the DIRECT algorithm, that could be adapted to address the challenge posed in the introduction. A brief description of SVM is included, necessary for the presentation in Section 3.

```
input : Design space $\mathscr{D}$, max_iteration (generation) $k$,
        sample (chromosome) size $l$
output: Solution $\mathbf{x}$

Initialization;
$\{\mathbf{x}_i\}_l \leftarrow$ Randomize ($\mathscr{D}$); // generate random samples
$\{f_i\}_l \leftarrow f(\{\mathbf{x}_i\}_l)$; // evaluate fitnesses of the samples
for $i \leftarrow 1$ to $k$ do
    $\{Elite\_kid, Cross\_kid, Mutation\_kid\} \leftarrow$
    Parentselection ($\{f_i\}_l, \{\mathbf{x}_i\}_l$);
    $Cross\_kid \leftarrow$ Crossover ($Cross\_kid$);
    $Mutation\_kid \leftarrow$ Mutation ($Mutation\_kid$, $\mathscr{D}$);
    $\{\mathbf{x}_i\}_l \leftarrow \{Elite\_kid, Cross\_kid, Mutation\_kid\}$;
    $\{f_i\}_l \leftarrow f(\{\mathbf{x}_i\}_l)$;
end
$\mathbf{x} \leftarrow$ Solution ($\{\mathbf{x}_i\}_l$, $\{f_i\}_l$); // pick one of the best
```

**FIGURE 1**. GENETIC ALGORITHM PSEUDO-CODE.

### Derivative-free optimization algorithms

Although not designed for the task described in this paper, several DFO algorithms may be adjusted to cope with binary class outputs. These include GAs and DIRECT method [14] [15] since both can handle set-in set-out evaluations and their number of function calls can be kept the same once the iteration numbers and sampling sizes are set to be the same. We shall mention that while pattern search and simplex algorithms may also be adaptable [13], no implementation for a set-in set-out black-box model is realized at this point. One may refer to [16] for a full review on these and other DFO methods. Here we briefly introduce the GA and DIRECT concepts and provide their pseudo-codes suitable for binary classes outputs.

**Genetic algorithm** A generational GA from MATLAB [17] is adopted containing parent selection, crossover and mutation sub-routines. Sample points $\mathbf{x}$ are used directly as real-valued chromosomes, and their fitness is associated with their class labels. Points with higher (lower) function values are labelled as “1” (“-1”)’s and have higher (lower) fitnesses. The parent selection routine generates three types of parents from the current generation: Elite ones are those with the highest fitnesses in the generation, while crossover and mutation children are selected for these two operations. The portions of these three types are fixed during the process. The termination criterion is not specified here, since the GA always runs to its maximum iteration in the experiment. Figure 1 has this pseudo-code.

**The DIRECT algorithm** The DIRECT algorithm treats a bounded design space as a hypercube and divides this hypercube iteratively to search for global optima. The dividing scheme is designed so that both hypercubes with large volumes and those

```
input  : Design space $\mathscr{D}$, max_iteration $k$
output: Solution $\mathbf{x}$
Initialization;
$Q \leftarrow \{\}$; // leave space for memory matrix Q. Q keeps
  record of hypercube information in each iteration
$i \leftarrow 0$; // iteration counter
$id \leftarrow 1$; // current active hypercube id
while $i \leq k$ do
    Divide current active box and evaluate;
    $Q \leftarrow$ Divide ($Q$, $id$, $i$, $\mathscr{D}$);
    Select active hypercube;
    $id \leftarrow$ Select ($Q$, $i$);
    In case more than one hypercube is selected, make an
    extra comparison between the multiple hypercube
    centers and pick the best one;
    while $||id|| > 1$ do
        $id \leftarrow$ Compare ($Q$, $id$);
        $i \leftarrow i+1$;
    end
    $i \leftarrow i+1$;
end
$\mathbf{x} \leftarrow$ Solution ($Q$, $k$, $id$);
```

**FIGURE 2**. DIRECT ALGORITHM PSEUDO-CODE.

with high (or low in minimization) function values will have the priority to be divided. The original deterministic DIRECT algorithm cuts a hypercube from all of its dimensions when its sides are of the same length. This cut varies the sampling size and makes it difficult to compare algorithms. Therefore, we modified the scheme to randomly choose one side for division under this situation. The termination criterion is not specified for the same reason as for the GA. Figure 2 shows the pseudo-code for this modified DIRECT.

## Support vector machine

Support vector machines were first introduced by Vapnik et al. in 1982 [18] as a classification and regression tool [19]. In this study we consider SVM only in a binary classification setting. Formally, SVM maximizes the margin between the two classes $\{(\mathbf{x}_i, y_i)\}$, $i = 1, ..., l$ where $\{\mathbf{x}_i\}_l$ are the data points with $n$ dimensions and $\{y_i\}_l$ are the corresponding binary labels. In their simplest form, SVMs are hyperplanes separating the two classes, with $y_i = 1$ on one side and $y_i = -1$ on the other. Data points closest to the hyperplanes are called support vectors. In general, such a hyperplane may not be found in the design space, and thus SVM maps the points $x_i$ from the design variable space $\mathscr{D}$ to a "feature" space $\mathscr{F}$ of some dimension $N$ (can be infinite):

$$\mathbf{x} = (x_1, ..., x_n) \mapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_N(\mathbf{x})), \tag{1}$$

where a hyperplane can be found. The resulting decision boundary in the input space has the form:

$$g(\mathbf{x}) := <\mathbf{w}, \phi(\mathbf{x})> + b = 0, \tag{2}$$

where $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)$ when the margin between the two classes in $\mathscr{F}$ , which is proportional to $1/||\mathbf{w}||^2$, is to be maximized. We obtain the $\alpha$'s by solving the following dual form of the optimization problem that achieves maximum margin (the primal is omitted here for brevity):

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x_i}, \mathbf{x_j}), \qquad (3) \\ s.t. \quad & \sum_{i=1}^{l} y_i \alpha_i = 0, \\ & \alpha_i \geq 0, \; i = 1, ..., l, \\ & \text{where } K(\mathbf{x_i}, \mathbf{x_j}) = <\phi(\mathbf{x_i}), \phi(\mathbf{x_j})> . \end{aligned}$$

$K(\mathbf{x_i}, \mathbf{x})$ is called a Mercer kernel operator. By introducing a kernel, we can find the decision boundary without knowing the form and dimensionality of $\phi$:

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x_i}, \mathbf{x}) + b = 0. \tag{4}$$

Commonly used kernels include the polynomial kernel $K(\mathbf{x}, \mathbf{z}) = (<\mathbf{x}, \mathbf{z}> + 1)^d$ and the radial basis function kernel $K(\mathbf{x}, \mathbf{z}) = exp(-\gamma||\mathbf{x} - \mathbf{z}||^2)$, for $\gamma > 0$. The latter is used in this study since it is reported to have good performance in most cases and is recommended as a default choice [20].

## SVM SEARCH ALGORITHM

Let the initial design space be $\mathscr{D}_0$ and that of iteration $i$ be $\mathscr{D}_i$, where $i = 1, ..., k$. The algorithm starts with $l_1$ random samples in $\mathscr{D}_0$ using a Latin Hypercube design. 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 will be drawn from the region $g_1(\mathbf{x}) > 0$ using a max-min sampling strategy and the user will be asked to label the combined set of these new samples and the $m$

Copyright © 2010 by ASME

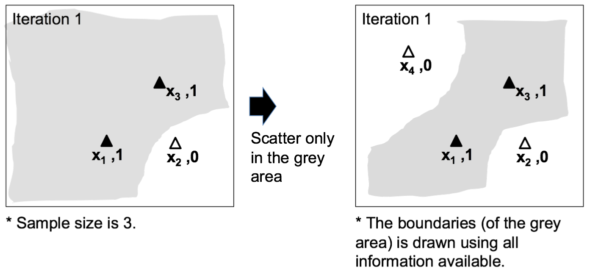
Iteration 1
$x_3$ ,1
$x_1$ ,1
$x_2$ ,0
* Sample size is 3.
Scatter only in the grey area
Iteration 1
$x_4$ ,0
$x_3$ ,1
$x_1$ ,1
$x_2$ ,0
* The boundaries (of the grey area) is drawn using all information available.

**FIGURE 3**. SVM SEARCH SCATTERING AND CLASSIFICATION

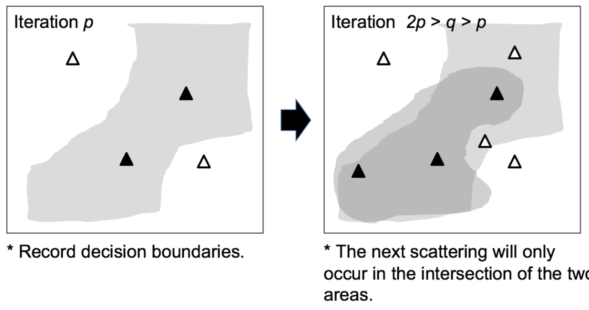
Iteration $p$
* Record decision boundaries.
Iteration $2p > q > p$
* The next scattering will only occur in the intersection of the two areas.

**FIGURE 4**. SVM SEARCH DESIGN SPACE REDUCTION

"winners" from the first stage. Once new labels are assigned and old labels are updated, a new decision boundary $g_2(\mathbf{x}) = 0$ will be generated upon the current total data set $\{\mathbf{x}_i, y_i\}_{2l_1-m}$. We illustrate this process in Fig. 3. It should be noted that samples with label "$-1$" need no further update. To speed up convergence, the decision boundary is recorded after each $p$ iterations, where $p$ is an adjustable parameter. Any further decision boundaries drawn will be combined with the recorded one in determining the future sampling space. This space reduction is illustrated in Fig. 4. One shall consider this as a way to speed up convergence at the risk of missing the optima. The algorithm terminates when the maximum number of iterations is reached or the design space constrained by the decision boundaries is "small enough". The pseudo code of this algorithm is shown in Fig. 5.

Several subroutines in the algorithm need clarification. The "Scatter" function (Fig. 6) generates $l$ samples in the design space constrained by the decision boundary set. Each sample is generated so that its minimum distance to all the other labeled data is maximized. "Label" (Fig. 7) labels the input data set. For preference elicitation purposes, this routine will require human evaluation. However, as an early test of the algorithm, we use functions to mimic human evaluation. In this sense, "Label" clusters function values of the sample set and labels them into two classes. The function "SVM" calls an SVM package "libsvm" [20], and *svm_options* stores parameters required to perform SVM. In this study, parameters are set to the defaults given in the cited source, the weight $C$ [20] is set at $1e6$ and no parameter tuning is performed. The "SmallEnough" routine checks whether there are still points lying in the constrained space using a fixed mesh of the design space. It should be noted that the current "Scatter" routine is computationally inefficient and requires improvement.

```
input  : Design space $\mathscr{D}$, max_iteration $k$, max_sample $l$
output: Decision boundary set $\{g(\mathbf{x}) = 0\}$
Initialization;
$g\_set \leftarrow \{\}$; // initialize decision boundary set
$\{\mathbf{x}_j\}_l$ = Scatter ($g\_set, \mathscr{D}, l$); // scatter $l$ samples in $\mathscr{D}$
  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 $k$ do
    Check if it is time to record current decision boundaries;
    if $rec == p$ then $g\_set = g_{i-1}(\mathbf{x})$, $rec0$;
    else $rec = rec + 1$;
    Check termination criteria;
    if SmallEnough ($\{g\_set, g_{i-1}(\mathbf{x})\}$) then
        break;
    end
    $g_i(\mathbf{x})$ = 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, $\mathbf{X}$ is the total labeled sample set
    $\{y_j\}_l$ = Label ($\{\mathbf{x}_j\}_l$); // Update lables
end
```

**FIGURE 5**. SVM SEARCH ALGORITHM PSEUDO-CODE.

```
input  : Decision function set $g\_set$, Design space $\mathscr{D}$,
         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 ($\mathscr{D}$, $g\_set$, $\mathbf{X}$); // $\mathbf{x}'$ has the maximum
      minimal distance to $\mathbf{X}$ in $\mathscr{D}$ and $g\_set$
    $\mathbf{X} = \{\mathbf{X}, \mathbf{x}'\}$; // Update the total sample set
end
```

**FIGURE 6**. SCATTER SUB-ROUTINE FOR SVM SEARCH.

Copyright © 2010 by ASME

```
input  : Samples $\{\mathbf{x}_i\}_l$
output: Binary lables $\{y_i\}_l$

Function evaluation;
for $i = 1$ to $l$ do
    $f\_eva_i$ = PrefFUN ($\mathbf{x}_i$); // PrefFUN is a user defined function
end
$\{y_i\}_l$ = Clustering ($\{f\_eva_i\}_l$); // Clustering calls MATLAB k-means algorithm to perform clustering on $\{f\_eva_i\}_l$
```

**FIGURE 7**. SUB-ROUTINE TO MIMIC HUMAN EVALUATION.

## RESULTS

In this section we show both simulation results and experiments to support the proposed algorithm. In the simulation part, we first show that SVM search can successfully locate small regions with near optimal function values. Its performance is then compared with that of GA and DIRECT on simulation test functions including 2D Rosenbrock, Six-hump Camel back, Branin, Shubert (Eqn. (5) through (8) in the appendix) and 3D Gaussian. DIRECT is put out of the race at an early stage due to its inferior performance on both 2D and 3D cases. In the experiment part, we show how the proposed algorithm can be incorporated into a preference elicitation task.

## SIMULATION RESULTS

**SVM Search Performance** Since the SVM search algorithm only reduces the design space rather than locate points, we show here the final constrained design space (the intersection of all constraints) at the end of the last iteration. It should be noted that these results are not deterministic due to the stochastic nature of the outputs from the maximin and clustering sub-routines in Fig. 6 and 7. The maximum iteration number is set at 20 and the sample size in each iteration is 5. The record parameter $p = 5$.

Figure 8 to 10 shows single runs of the algorithm for Rosenbrock, Camelback and Branin functions, respectively. The contour represents the functions, and the solid curves are the decision boundaries. In each case, the boundary with a larger area is the previously recorded boundary and the other one is that of the last iteration. The resulting design space will be the intersection of these two. As a note on the results, the global optimal solution of Rosenbrock, $(1,1)$, is missed by the SVM search. This is because the algorithm aims to locate the design space with high function values but not necessarily near the global optimal solution(s). The design space is reduced to a small region around one of the two global optimal solutions in the Camelback case. In the Branin case, the final decision boundary set appears as a dot, close to one of the three global optimal solutions. Although experiments show that SVM search has stable and reliable performance on these continuous and smooth functions, the algorithm works poorly on noisy functions (function with a large number of local optima), as is shown in Fig. 11.

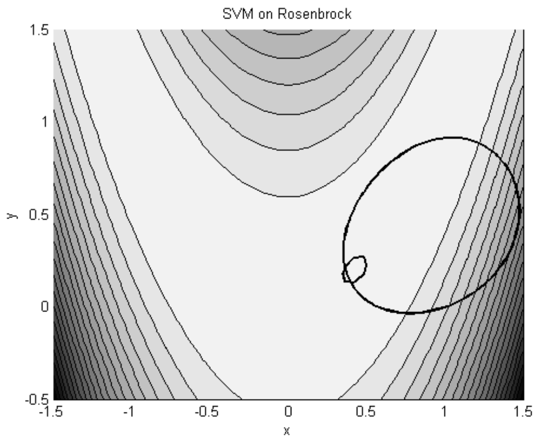

**FIGURE 8**. SVM SEARCH ON 2D ROSENBROCK FUNCTION.

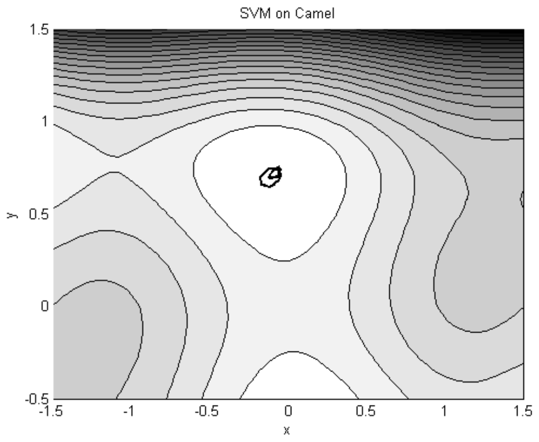

**FIGURE 9**. SVM SEARCH ON 2D SIX-HUMP CAMELBACK FUNCTION.

### Comparison between SVM search, GA and DIRECT

The performance of the proposed SVM search algorithm is compared with that of GA and DIRECT. We set the performance measure as the difference between a solution and the real optimum. Since SVM search only locates a region rather than a point, we use the sample point with the optimal function value as the final guess. In real experiments, this means that in the last iteration, instead of asking the user to pick a few samples from the set, we only request the best one. To handle the stochastic nature of these algorithms, we execute 10 runs of every experiment.

The first test shown in Table 1 compares performance of all

Copyright © 2010 by ASME

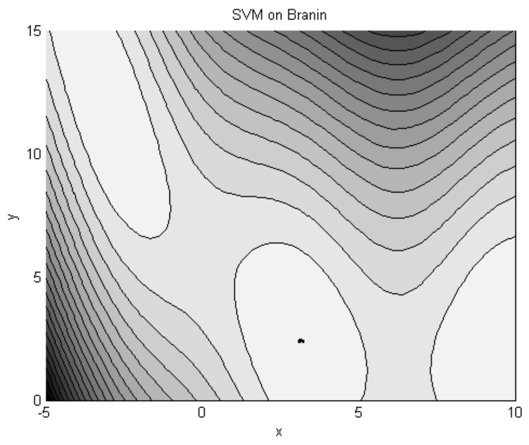

**FIGURE 10**. SVM SEARCH ON 2D BRANIN FUNCTION.

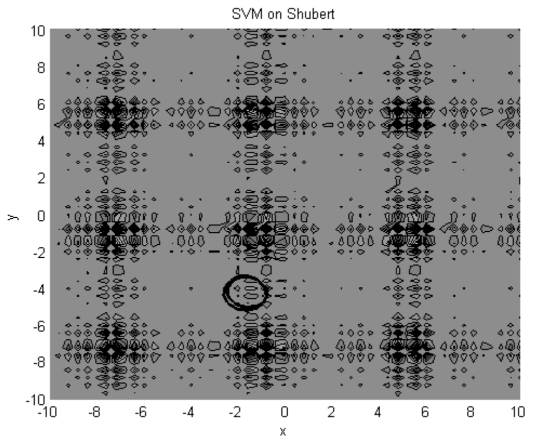

**FIGURE 11**. SVM SEARCH ON 2D SHUBERT FUNCTION.

**TABLE 1**. SVM SEARCH, GA AND DIRECT COMPARISONS

| Case | 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) | 31.03 (7.45) |
| DIRECT | 11.65 (21.01) | 0.44 (0.27) | 20.19 (0.56) |
| random | 19.82 (13.47) | 18.32 (18.35) | 22.95 (21.73) |

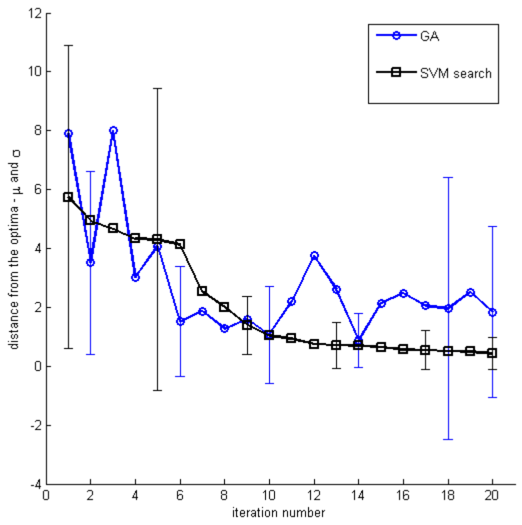

**FIGURE 12**. SVM SEARCH VS. GA ON ROSENBROCK FUNCTION.

three algorithms under maximum iteration of 20 and maximum sampling size of 3 per iteration, since DIRECT is designed to take only 3 samples per iteration. The values shown are means and standard deviations (in the bracket) of the gaps between the results of the algorithms and the global optima of the test functions, namely, 2D Rosenbrock, six-hump Camelback and 2D Branin. To further confirm whether these methods succeed or not, a random sampling algorithm is used as a baseline performance. This algorithm merely samples points randomly during the process, keeping better points and dropping worse points.

We decide to remove DIRECT from the competition due to its inferior performance. Besides, while the performance of SVM search and GA can be improved with an increase in the sample size as can be seen from Fig. 12 to 14 where the sample size is set to 5, DIRECT has no such flexibility. Fig. 12 to 14 also show consistent improvements in the SVM search with increasing iterations, while GA fails to share this property.

We are also interested in testing how dimensionality will affect both algorithms. A multidimensional Gaussian function $f(\mathbf{x}) = \exp(-\mathbf{x}^T\mathbf{x})$ is used for this purpose. Throughout the test, the sample size in each iteration is fixed at 5 and the maximum iteration number is 20. Comparison between SVM search and GA in Fig. 15 shows that the proposed SVM search algorithm suffers less than the GA from increasing dimensionality.

## EXPERIMENT RESULTS

The purpose of this section is to show how the proposed SVM search algorithm can be incorporated into a user-computer interaction environment and successfully elicit user preference through iterations.

Here we parameterize a 2D heart shape design with 3 variables. Before starting the experiment, we ask the user to specify a heart shape that he/she would like to search for. We scan and

Copyright © 2010 by ASME

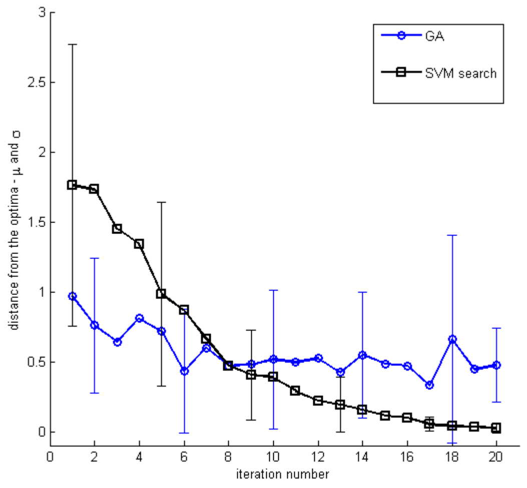

**FIGURE 13**. SVM SEARCH VS. GA ON SIX-HUMP CAMEL-BACK FUNCTION.

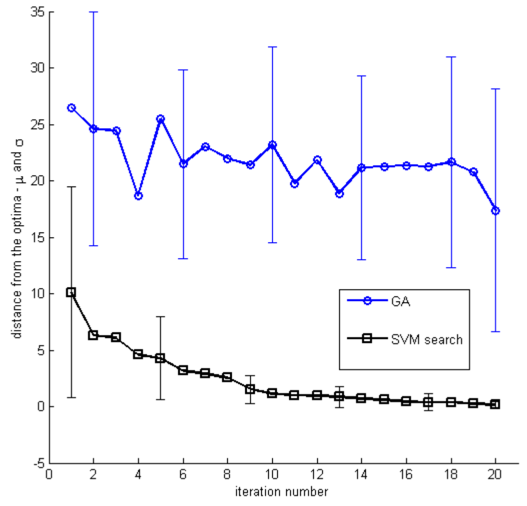

**FIGURE 14**. SVM SEARCH VS. GA ON BRANIN FUNCTION.

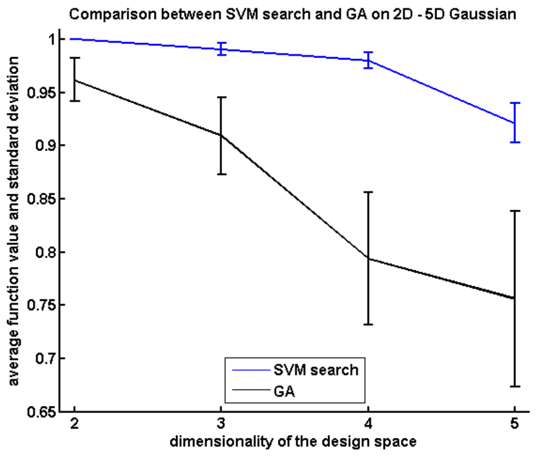

**FIGURE 15**. SVM SEARCH VS. GA WITH RESPECT TO DIMENSIONALITY.

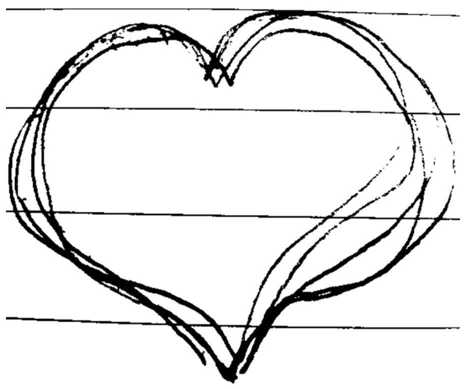

**FIGURE 16**. SCAN OF A USER TARGET DESIGN.

show this target design in Fig. 16. The user is then asked to pick the heart shapes that are closer to the target during the interaction. Figure 17 shows the sequence of this interaction where 6 shapes are presented in each iteration and the user's choices are highlighted. One shall notice that in early stages of the interaction, the algorithm tries to explore the design space based on the maximin routine in Fig. 6. Therefore the presented designs tend to be on the boundaries of the preferred design space as seen in Iteration 3, while in later iterations, the preferred design space is reduced to the extent that all samples become similar to each other and the interaction converges.

The experiment is very simple in that we can easily sketch the heart shape, leaving no chance for the interaction to be useful. However, the idea is to apply the proposed algorithm on eliciting design preference from complex designs such as vehicle exterior shapes. Such a parametric 3D exterior design tool can be found in [21] and Fig. 18 shows the variation of this design task. A comparison between the proposed SVM search algorithm with the traditional interaction genetic algorithm will need to be studied with the help of sufficient user feedback.

## CONCLUSION

We attempted to address the design preference elicitation problem from a derivative-free optimization perspective. We examined how a DFO problem with binary class outputs can be solved by three different algorithms, the modified GA and DIRECT algorithms and a new SVM search algorithm. We com-

Copyright © 2010 by ASME

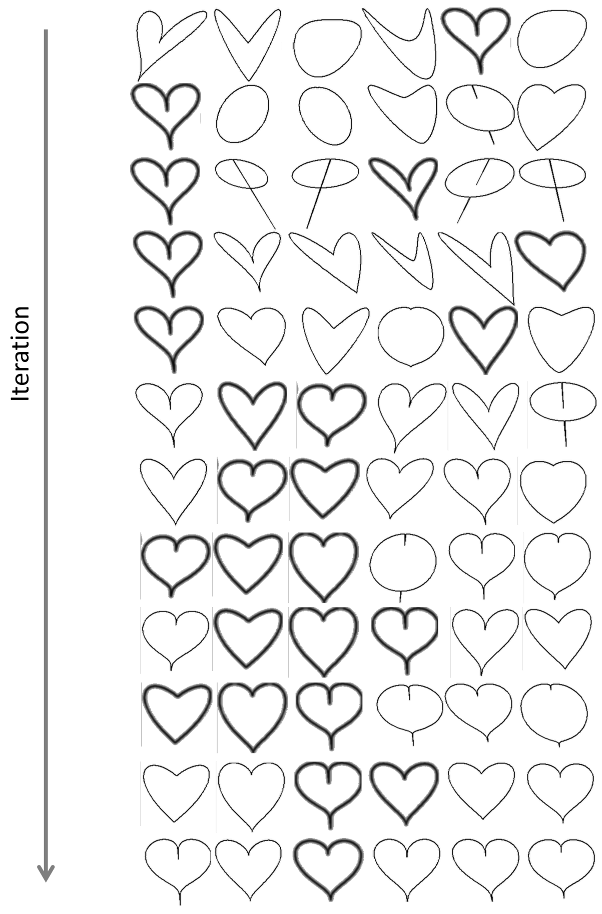

**FIGURE 17**. EXPERIMENT ON HEART SHAPE SELECTION.

pared algorithmic performance for a few classic test functions looking at keeping iteration number and sample size small, and exploring how problem dimensionality affects performance. The SVM search stood out as a preferred search method for these simulations. We also show through a pilot experiment that the proposed algorithm can be incorporated into real-time user computer interaction for design preference elicitation purpose. There are several limitations in this work. We have no theoretical basis yet to support the performance and robustness of the proposed algorithm. It is also necessary to execute sufficient user interaction studies on complex design tasks.

## ACKNOWLEDGEMENT

This research was partially supported by a grant from Ford Motor Company under the Ford-University of Michigan Innovation Alliance, and by the Automotive Research Center, a US Army Center of Excellence headquartered at the University of Michigan. This support is gratefully acknowledged.

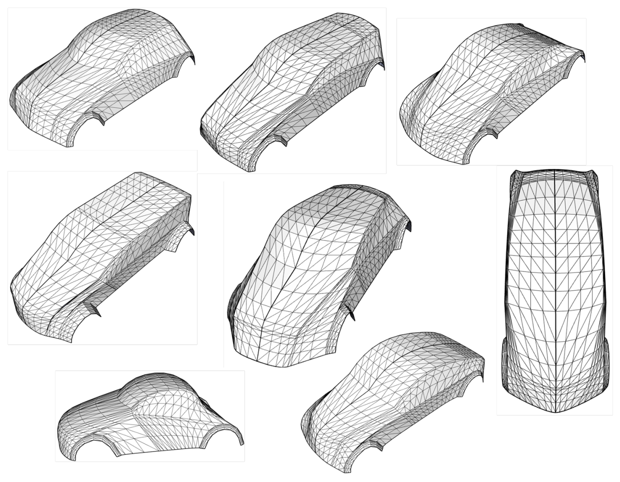

**FIGURE 18**. VEHICLE EXTERIOR DESIGN GENERATOR TO BE INCORPORATED.

## REFERENCES

[1] Green, P., and Srinivasan, V., 1978. "Conjoint analysis in consumer research: issues and outlook". *Journal of consumer research,* ***5***(2), p. 103.

[2] Nagamachi, M., 1995. "Kansei Engineering: A new ergonomic consumer-oriented technology for product development". *International Journal of Industrial Ergonomics,* ***15***(1), pp. 3–12.

[3] Wittink, D., and Cattin, P., 1989. "Commercial use of conjoint analysis: An update". *The Journal of Marketing,* ***53***(3), pp. 91–96.

[4] Jindo, T., and Hirasago, K., 1997. "Application studies to car interior of Kansei engineering". *International Journal of Industrial Ergonomics,* ***19***(2), pp. 105–114.

[5] Diaper, D., ed., 1989. *Knowledge elicitation: principle, techniques and applications*. Springer-Verlag New York, Inc., New York, NY, USA.

[6] Boy, G., 1991. *Intelligent assistant systems*. Academic Press.

[7] Gaver, W., Beaver, J., and Benford, S., 2003. "Ambiguity as a resource for design". In Proceedings of the SIGCHI conference on Human factors in computing systems, ACM New York, NY, USA, pp. 233–240.

[8] Chen, L., and Pu, P., 2004. "Survey of preference elicitation methods". *Swiss Federal Institute of Technology in Lausanne, Technical Report No. IC/200467*.

[9] 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.

[10] Sims, K., 1991. "Artificial evolution for computer graphics". *Computer Graphics,* ***25***(4), pp. 319–328.

[11] Tokui, N., and Iba, H., 2000. "Music composition with

Copyright © 2010 by ASME

interactive evolutionary computation". In GA2000. Proceedings of the third International Conference on Generative Art.
[12] Kim, H., and Cho, S., 2000. "Application of interactive genetic algorithm to fashion design". *Engineering Applications of Artificial Intelligence,* ***13***(6), pp. 635–644.
[13] 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.
[14] Jones, D., 2001. "The DIRECT global optimization algorithm". *Encyclopedia of optimization,* ***1***, pp. 431–440.
[15] Finkel, D., 2003. "DIRECT optimization algorithm user guide". *Center for Research in Scientific Computation, North Carolina State University*.
[16] Rios, L., 2009. "Algorithms for derivative-free optimization". PhD Thesis, University of Illinois at Urbana-Champaign.
[17] Chipperfield, A., and Fleming, P., 1995. "The MATLAB genetic algorithm toolbox". In Colloquium Digest-IEE, Citeseer, pp. 10–10.
[18] Vapnik, V., 1982. "Estimation of dependences based on empirical data". *NY: Springer-Verlag*.
[19] Drucker, H., Burges, C., Kaufman, L., Smola, A., and Vapnik, V., 1997. "Support vector regression machines". *Advances in neural information processing systems*, pp. 155–161.
[20] Chang, C., and Lin, C., 2001. LIBSVM: a library for support vector machines.
[21] Yi, R., 2009. "An interactive modeling environment for automotive exterior design". MS Thesis, University of Michigan, Ann Arbor.

## APPENDIX: TEST FUNCTIONS

2D Rosenbrock function:

$$f(x,y) = -\left((1-x)^2 + 100(y-x^2)^2\right)$$
$$\mathscr{D} = \{(x,y),\ x \in [-1.5, 1.5],\ y \in [-0.5, 1.5]\} \tag{5}$$

2D Six-hump Camelback function:

$$f(x,y) = -\left((4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4+4y^2)y^2\right)$$
$$\mathscr{D} = \{(x,y),\ x \in [-3, 2],\ y \in [-3, 2]\} \tag{6}$$

2D Branin function:

$$f(x,y) = -\left((y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x) + 10\right)$$
$$\mathscr{D} = \{(x,y),\ x \in [-5, 10],\ y \in [-5, 10]\} \tag{7}$$

2D Shubert function:

$$f(x,y) = -\left(\sum_{j=1}^{5} j\cos\left((j+1)x_1 + j\right)\right)\left(\sum_{j=1}^{5} j\cos\left((j+1)x_2 + j\right)\right)$$
$$\mathscr{D} = \{(x,y),\ x \in [-10, 10],\ y \in [-10, 10]\} \tag{8}$$

Copyright © 2010 by ASME