Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2012
August 12-15, 2012, Chicago, United States

DETC2012-70605

ON DESIGN PREFERENCE ELICITATION WITH CROWD IMPLICIT FEEDBACK

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

We define preference elicitation as an interaction, consisting of a sequence of computer queries and human implicit feedback (binary choices), from which the user's most preferred design can be elicited. The difficulty of this problem is that, while a human-computer interaction must be short to be effective, query algorithms usually require lengthy interactions to perform well. We address this problem in two steps. A black-box optimization approach is introduced: The query algorithm retrieves and updates a user preference model during the interaction and creates the next query containing designs that are both likely to be preferred and different from existing ones. Next, a heuristic based on accumulated elicitations from previous users is employed to shorten the current elicitation by querying preferred designs from previous users (the "crowd") who share similar preferences to the current one.

1 Introduction

"What do people really like?" Getting an answer for this question is the holy grail in the long pursuit of consumer preference, and has attracted strong interest from the engineering design community. In this work, we call the problem of finding the most preferred design for a user *preference elicitation*. We use the term *user* for anyone whose preference is of interest and the term *design* for products or services that can be tailored to meet user preferences.

Marketing researchers starting in the 1970s introduced conjoint analysis [17] to estimate the preference of a population as a function of product attributes based on either consumer purchase data or survey responses. Conjoint analysis has been reportedly successful in the engineering design community [7,13,14,18,28], but it has also been questionable, as the theoretical assumptions may not always hold [6, 15]. In addition, modeling consumers' preferences is different from identifying the most preferred design by a user. While the former can be treated as an estimation problem, the latter is more an optimization one.

Computer science research took a more direct approach by introducing Interactive Evolutionary Computation (IEC) [9, 11, 24, 25, 27]. This category of research considers the preference of a consumer as a black-box function and optimizes it through evolution using Genetic Algorithms [11,12] or Genetic Programming [9,24]. During an IEC interaction, the computer presents a *query*, i.e., a set of designs from where the user chooses her relatively preferred ones. This binary comparison feedback, called *implicit feedback* [8], is then parsed by the computer into fitness of these designs, and a new query is created based on evolutionary operations, such as parent selection, crossover and mutation, on the existing designs. This completes one *iteration* of IEC. While IEC allows users to freely explore design possibilities, convergence is doubtful due to human fatigue in long-term interactions [27]. To circumvent this issue, Collaborative Interactive Evolution (CIE) was introduced. Built upon the evolution-based interaction capability of IEC, CIE incorporates crowd sourcing so that a design preferred by a group of people can evolve

[*]Address all correspondence to this author.

Copyright © 2012 by ASME

through a sequence of human-computer interactions [23,26]. The convergence issue still remains due to the arbitrary nature of evolutionary operations. In fact, few established criteria exist on how these operations will be tailored for a specific problem.

To this end, we previously introduced a query algorithm for preference elicitation as an analogy to the response surface method prevalent in solving black-box optimization problems [21]: In each iteration a user preference model is retrieved by classifying preferred designs against not-preferred designs from the accumulated feedback. Designs for the next query will then be sought that have high preference value based on this model and also be away from existing designs. This query algorithm based on learning (classification) is shown to outperform evolutionary computation in simulated tests in most cases [20].

As a continuation to our previous work, this paper aims to further enhance the convergence performance of preference elicitation by two improvements: (1) We first realize that the implicit feedback from the user has richer information than merely binary labels on designs. The feedback represents a comparison graph where each connected vertex pair shows the result of a comparison. We show empirically that exploiting the information from this comparison graph will help to retrieve a user preference model during the interaction[1]. (2) We consider how queries can be further shortened when a large amount of preference elicitations are recorded from a crowd. Various heuristic query algorithms are investigated to this end. Simulation results show that when preferences of the crowd form clusters, heuristics based on crowd information can successfully reduce the average query size needed to elicit their preferences.

The rest of this paper is organized as follows: Section 2 introduces the query algorithm for preference elicitation and compares the proposed methods against existing ones through simulated tests. Section 3 investigates how queries can be further shortened with the help of previous elicitation experiences. We then provide a review of related work in Section 4 and conclude in Section 5.

2 Preference Elicitation from Individuals

This study assumes that each user under preference elicitation acquires a preference function $f : \mathscr{X}_c \to \mathbb{R}$, where $\mathscr{X}_c$ is a pre-defined candidate set containing designs represented as vectors. The objective of elicitation is to locate $\mathbf{x}^* \in \mathscr{X}_c$ that has the maximal preference value within a limited number of iterations, called the optimally-preferred design. We consider elicitation as a black-box optimization problem where the outputs are implicit feedback, with a further assumption that the preference function is fixed throughout the elicitation. In other words, the user's preference is not affected by the presence of queries. Although this assumption deserves inspection [16, 19], it is widely adopted by most of the marketing and preference retrieval research [3, 17]. This section introduces the proposed query algorithm for eliciting individuals' optimally-preferred designs. The algorithm is developed based on the Efficient Global Optimization (EGO) [10] algorithm originally designed for black-box optimization problems, thus is called *EGO Query*.

[1]In this work, we use the terms "interaction" and "elicitation" interchangeably.

2.1 Preference retrieval with implicit feedback

We formulate a preference retrieval problem for the given sample set $\mathbf{X}$ ($n \times p$) with n samples from $\mathscr{X}_c$, and the implicit feedback $\mathbf{G}$ ($n \times n$) where $\mathbf{G}_{ij} = 1$ if and only if sample i is more preferred than sample j according to the user. We introduce a retrieval function

$$\hat{f}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}), \tag{1}$$

where $\mathbf{w}$ ($n \times 1$) is the user profile and $\Phi_i(\mathbf{x})$ is defined as the similarity between $\mathbf{x}$ and the ith sample $\mathbf{x}_i$ in $\mathbf{X}$:

$$\Phi_i(\mathbf{x}) = \exp(-\lambda||\mathbf{x}_i - \mathbf{x}||^2), \tag{2}$$

with the spread λ set to $1/p$. From another perspective, $\Phi(\mathbf{x})$ is a nonlinear mapping of $\mathbf{x}$ to a feature space.

We define the optimal estimator $\hat{\mathbf{w}}$ as the solution to the convex problem in Equation (3) where C is a weighting parameter on the slack variables ξ, representing the importance of training error:

$$\begin{array}{lll} (P1) & \underset{\hat{\mathbf{w}}}{\text{minimize}} & \frac{1}{2}\hat{\mathbf{w}}^T\hat{\mathbf{w}} + C\sum \xi_{ij} \\ & \text{subject to} & \mathbf{w}^T\left(\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\right) \geq 1 - \xi_{ij}, \\ & & \xi_{ij} \geq 0, \\ & & \forall i,j \text{ such that } \mathbf{G}_{ij} = 1. \end{array} \tag{3}$$

The solution to Equation (3) is derived equivalently from its dual problem in Equation (4) where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is a kernel matrix with elements $Q_{ab} =< \Phi(\mathbf{x}_{i_a}) - \Phi(\mathbf{x}_{j_a}), \Phi(\mathbf{x}_{i_b}) - \Phi(\mathbf{x}_{j_b}) >$. Throughout this paper, the inner product $< \cdot,\cdot >$ is implemented using a radial basis with spread $\lambda = 1/p$, i.e., $Q_{ab} = \exp(-\lambda||(\mathbf{x}_{i_a} - \mathbf{x}_{i_b})||_2^2) + \exp(-\lambda||(\mathbf{x}_{j_a} - \mathbf{x}_{j_b})||_2^2) - \exp(-\lambda||(\mathbf{x}_{i_a} - \mathbf{x}_{j_b})||_2^2) - \exp(-\lambda||(\mathbf{x}_{j_a} - \mathbf{x}_{i_b})||_2^2)$. We also assume that human choices are consistent with their preferences, i.e., a user that prefers design A over design B will choose A over B, in which case the weight C is set to a high value ($C = 10^6$) in all simulated tests to avoid large slacks.

$$(D1) \quad \underset{\alpha}{\text{minimize}} \quad \frac{1}{2}\alpha^T \mathbf{Q}\alpha - \mathbf{1}^T\alpha \qquad \text{subject to} \quad \mathbf{0} \leq \alpha \leq C\mathbf{1}. \tag{4}$$

An alternative kernel definition applied has the form $Q_{ab} = \exp(-\lambda||(\mathbf{x}_{i_a} - \mathbf{x}_{j_a}) - (\mathbf{x}_{i_b} - \mathbf{x}_{j_b})||_2^2)$. We call retrieval with this kernel $(P2)$ and show that although the kernel is not strictly correct in theory, it leads to a better convergence performance of EGO Query in various simulated tests when query sizes are small. This further leads to a hybrid query method which outperforms both $(P1)$ and $(P2)$.

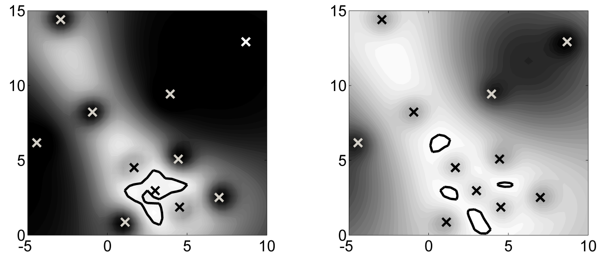
2.2 Balancing exploitation and exploration in a query

A new query $\mathbf{x}_{n+1}$ is made once the retrieval is updated with the current data ($\mathbf{X}$ and $\mathbf{G}$) to potentially get closer to the optimally-preferred design. Such a query can be found by simultaneously exploiting the retrieval function and exploring the candidate set $\mathscr{X}_c$. In EGO research, a query is found by optimizing a merit function defined on $\mathscr{X}_c$ that combines the predictive response surface from current observations and the uncertainty in the prediction [10]. Although not theoretically proven, it is recommended that queries at early stages shall focus more on exploration while those at later stages on exploitation [22].

A variety of merit functions in literature follows this principle, including expected improvement [10] and lower confident bound [5]. It is, however, acknowledged that these merit functions are usually highly nonlinear over $\mathscr{X}_c$. As an example, contours of merit functions after a few iterations in a simulated elicitation process are illustrated in Figure 1 where brighter areas indicate higher merit function values. The crosses represent queried designs, and are colored only to be differentiated from the background. Notice that regions with the highest merit function values (bright areas circled out) are irregular, causing the region to have multiple local optima and requiring a heuristic search (branch-and-bound or genetic algorithm) to find its global optimum. Considering that the EGO Query algorithm will gradually shift its focus from exploration to exploitation, a fairly large amount of designs will be scattered in a small region close to the optimal solution in later iterations, which could intensify the nonlinearity of the merit functions and cause searches to be inefficient and impractical for real-time human-computer interactions.

To resolve this computational cost issue, we introduce the following merit function defined on $\mathscr{X}_c \times \mathbb{R}$:

$$f_{\text{merit}}(d, \mathbf{x}) = w_1\hat{f}(\mathbf{x}) + w_2 d - \sum_{i=1}^{n} \exp(d - ||\mathbf{x} - \mathbf{x}_i||_2^2). \tag{5}$$



(a) Expected improvement (b) Lower confident bounding

FIGURE 1. Merit functions during the query process on a Branin function, see Equation (7).

The variable d represents the minimum distance from $\mathbf{x}$ to all existing samples $\mathbf{x}_i$ for $i = 1, ..., n$. This merit function follows the goal of balancing exploration and exploitation: It tries to maximize the retrieval function $\hat{f}(\mathbf{x})$ and the range of exploration measured by d. The exponential penalty term in Equation (5) is a numerical treatment to push the value of d close to its physical meaning. The setting of the weights w_1 and w_2 determines the focus on either exploration or exploitation and shall be calibrated according to the scale of the two objectives. In this study we set

$$w_1 = \frac{\sum_{i=1}^{p}(x_{\min}^i - x_{\max}^i)^2}{\hat{f}_{\max}}, \tag{6}$$

where $x_{\min}^i$ and $x_{\max}^i$ are the lower and upper bounds of the ith dimension of a design. The numerator of w_1 represents the longest distance in $\mathscr{X}_c$ and its denominator $\hat{f}_{\max}$ the optimum of $\hat{f}$; w_2 is set at 1 for the first half of iterations during an elicitation, and is reduced to 0.01 afterwards. This setting ensures the two objectives to have values of the same scale at early stages of the query process and shifts the focus to exploitation to improve convergence at later stages.

2.3 Simulated test results

In this subsection we test the convergence performance of EGO Query on minimizing a set of testing functions simulating human preferences. The testing functions include Branin and Six-hump Camelback functions of two dimensions and Gaussian functions of 10, 20 and 30 dimensions, see Equations (7) to (9). We set up the Gaussian functions to be difficult to optimize by specifying a small spread value and biasing the optimal solutions to one corner of $\mathscr{X}_c$.

Copyright © 2012 by ASME

Branin:

$$f(x,y) = (y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x) + 10,$$
$$\mathscr{X}_c = \{x \in [-5,10],\ y \in [-5,10]\}; \qquad (7)$$

Six-hump Camelback:

$$f(x,y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2,$$
$$\mathscr{X}_c = \{x \in [-3,2],\ y \in [-3,2]\}; \qquad (8)$$

Gaussian:

$$f(\mathbf{x}) = -\exp(-5\sum_{i=1}^{N}(x_i - 0.9)^2),$$
$$\mathscr{X}_c = \{x_i \in [-1,1]\ \forall i = 1,...,N\},\ N = 10,20,30. \qquad (9)$$

Retrievals by $(P1)$ and $(P2)$ are compared. In addition, we test a hybrid method that retrieves preference using $(P2)$ for the first half of the interaction and then switches to $(P1)$. The merit function applied on these three retrieval configurations follows Equation (5). The MATLAB genetic algorithm toolbox [4] is used to optimize the merit function. Two baseline query algorithms are added for comparison. We first introduce a query algorithm that represents an interactive genetic algorithm in a human-computer interaction, referred to as "GA". It uses the hybrid method for function retrieval and applies genetic algorithm directly on the retrieved values at sampled designs. The second baseline algorithm is introduced in [21], which considers user feedback as binary classes, i.e., preferred designs against not-preferred ones and the retrieval is thus based on a binary classification. This method is referred to as "Classification". All tests are repeated 20 times due to the random nature of genetic algorithms in solving the merit function. Errors at each iteration are measured as the gaps between the global optima and the minimal objective value found up to that iteration. Figure 2 shows the test results. Notice that the performances of "GA" and "Classification" are overlapping in Gaussian tests with 20 and 30 dimensions since they have negligible improvements within the given query size.

One interesting finding from these results is that retrieval by $(P1)$ is not a good choice in short runs, but it outperforms retrieval by $(P2)$ in a long run in most cases. Our explanation to this finding is that while $(P2)$ violates user feedback, it may have a better generalization error when observation is limited. Retrieval by $(P1)$, on the other hand, exploits the limited observations and may lead to inefficient queries at early stages. The hybrid approach is inspired by this observation. We also see that the proposed query method outperforms the baseline algorithms by a large margin for these tests.

3 Heuristic Query with Crowd Elicitation Records

The individual elicitation process can be depicted as a path, consisting of a sequence of vertices, see Figure 3. Each vertex on a path contains queries made up to and feedback prior to this iteration; each directed edge from a vertex contains the feedback to the current query. We label the leaf vertex of a path with the optimally-preferred design found for the user. Let the root vertex contain the same initial guesses for all individuals. The paths recorded from a crowd form a directed graph with the same root vertex, called the elicitation graph as shown in Figure 3. We attach a vector of costs $\mathbf{p}_i$ to the corresponding leaf vertex v_i. The elements in $\mathbf{p}_i$ represent the various numbers of queries that have been made to reach v_i in all previous elicitations. Denote $\bar{p}_i$ as the average cost of vertex v_i. Both $\mathbf{p}_i$ and $\bar{p}_i$ for all existing vertices in the graph are updated after some elicitations are recorded.

This section investigates how query strategies can be updated based on the cumulative elicitation records from a crowd. The intuition is that when optimally-preferred designs from the crowd are clustered in $\mathscr{X}_c$, an interaction can be shortened if we observe similarity between the current user and some previous user(s). The heuristic search works as follows: The hybrid EGO Query algorithm will be used as the default query method. After every m iterations with the default query, a heuristic query is made based on the retrieved user preference. This query asks the user to compare his current preferred design and one of the optimally-preferred designs from previous users that have similar preferences. Regardless of the user's response, the default search algorithm is switched back on after this heuristic query.

In order to be concise and clear, we limit our discussion to pairwise comparison, i.e., every query contains only two designs and the user must pick one out of the two, leading to a simplified case where each new query contains one design.

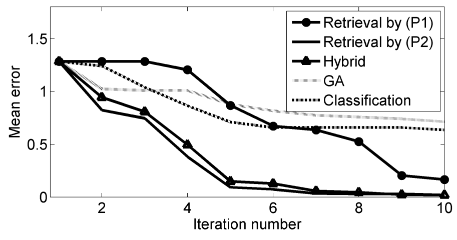
3.1 Heuristic query

To perform a heuristic query, we collect a set of optimally-preferred designs from previous users with similar preferences to the current one, denoted as the similar candidate set $\mathscr{X}_s = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$. The similarity between the current user and any previous user τ is measured as the angle between their profiles, $\hat{\mathbf{w}}_s$ and $\hat{\mathbf{w}}_{s\tau}$ correspondingly, at some iteration s:

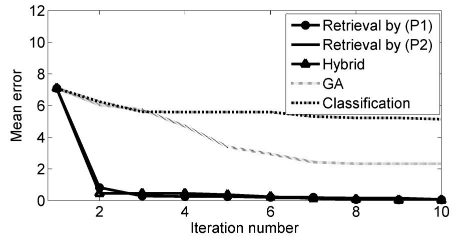
$$\text{similarity}_{s\tau} = \frac{\hat{\mathbf{w}}_s^T \hat{\mathbf{w}}_{s\tau}}{||\hat{\mathbf{w}}_s||_2 ||\hat{\mathbf{w}}_{s\tau}||_2}. \qquad (10)$$

We consider users similar to the current one as those who acquire the maximum similarity. Once $\mathscr{X}_s$ is established, a heuristic decision needs to be made on which design from $\mathscr{X}_s$ shall be picked

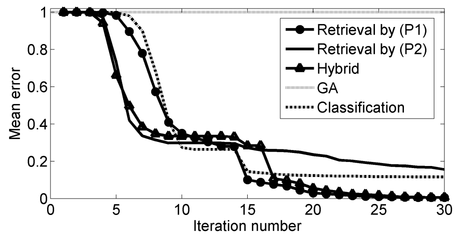
Copyright © 2012 by ASME



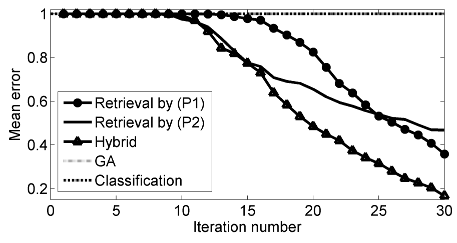
(a) 2D Six-Hump Camelback function



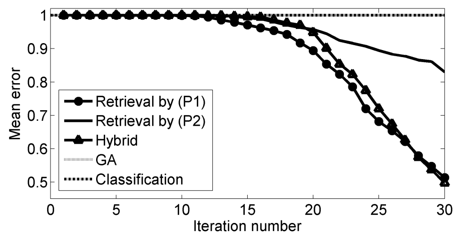
(b) 2D Branin function



(c) 10D Gaussian function



(d) 20D Gaussian function



(e) 30D Gaussian function

FIGURE 2. Convergence comparisons on testing functions.

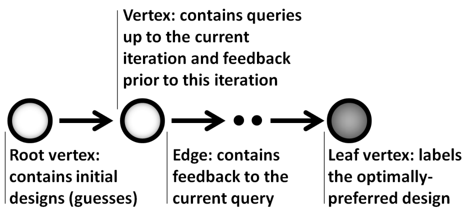

Vertex: contains queries up to the current iteration and feedback prior to this iteration
Root vertex: contains initial designs (guesses)
Edge: contains feedback to the current query
Leaf vertex: labels the optimally-preferred design

FIGURE 3. Individual elicitation recorded as a path

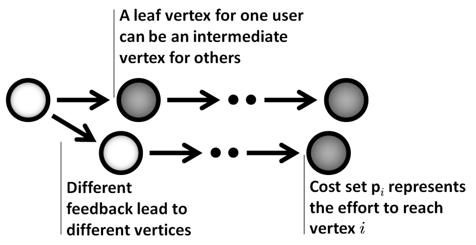

A leaf vertex for one user can be an intermediate vertex for others
Different feedback lead to different vertices
Cost set p_i represents the effort to reach vertex i

FIGURE 4. The elicitation graph formed from crowd paths

to present to the current user. Two heuristics are introduced below.

1. ($H1$) Design with the most potential gain: This heuristic picks the design that has the maximum average cost: $\mathbf{x}_{i^*}$ such that $\bar{p}_{i^*} = \max_i \bar{p}_i$, for $i = 1, ..., m$. The intuition here is that by querying the most costly preferred design from similar users, the query length can potentially be decreased the most.
2. ($H2$) Design with the least potential cost: Let $\mathscr{X}_i$ be a set of leaf vertices that are descendants of the ith element of $\mathscr{X}_s$. And let $\tilde{p}_i$ represent the averaged query cost when we choose to query $\mathbf{x}_i \in \mathscr{X}_s$ but it is not the optimally-preferred design for the user:

$$\tilde{p}_i = \frac{1}{|\mathscr{X}_i|} \sum_{j=1}^{|\mathscr{X}_i|} \bar{p}_{i_j}, \tag{11}$$

where $\mathbf{x}_{i_j} \in \mathscr{X}_i, \forall j$. The intuition here is to pick a candidate design that leads to the minimum extra query size needed when the heuristic query is not the optimally-preferred design.

3.2 Demonstration of $H1$ and $H2$

This subsection compares the performance of $H1$ and $H2$ on an experiment where simulated users interact with the computer in a sequence and the computer improves its query strategy ($H1$ or $H2$) based on the accumulated interaction records.

Copyright © 2012 by ASME

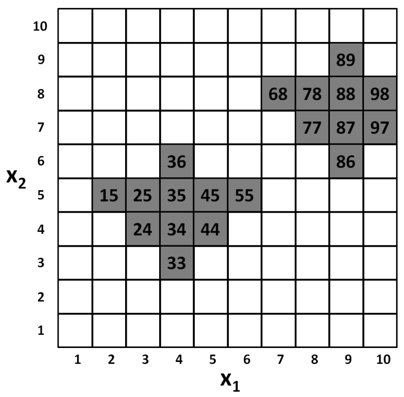


FIGURE 5. Setup of an experiment to test $H1$ and $H2$. A set of 100 candidate designs are represented by the blocks. The left bottom block is indexed as $\mathbf{x}_1$ and the top right one as $\mathbf{x}_{100}$. Optimally-preferred designs are uniformly drawn from the grayed blocks.

3.2.1 Simulated user setup

To start, we define $\mathscr{X}_c$ as a finite set on $\mathbb{R}^2$:

$$\mathscr{X}_c = \left\{\mathbf{x} \in \mathbb{R}^2, x_{1,2} \in \{0, 0.1, ..., 0.9\}\right\}, \tag{12}$$

with $|\mathscr{X}_c| = 100$. The experiment simulates 1000 users interacting with the computer in a sequence. To model the preferences of these simulated users, we manually define an optimally-preferred design set, and denote their indices as a set $\mathscr{I}_{opt}$, shown in Figure 5 as the grayed blocks. A random sequence $\mathbf{I}$ of size 1000 is generated uniformly from $\mathscr{I}_{opt}$ so that $\mathbf{x}_{I_i} \in \mathscr{X}_c$ represents the optimally-preferred design of user i. Accordingly, the preference of user i is modeled as

$$f_i(\mathbf{x}) = -\sum_{j=1}^{100} w_{ij} \exp(-||\mathbf{x} - \mathbf{x}_j||_2^2). \tag{13}$$

Here w_{ij} is the jth element of user i's profile, defined as

$$w_{ij} = \begin{cases} 1, & \text{when } j = I_i \\ 0, & \text{otherwise} \end{cases}. \tag{14}$$

The above experiment setup has these properties: (1) each user acquires a unimodal preference function where the optimal solution belongs to $\mathscr{X}_c$, and (2) the optimally-preferred designs are clustered.

3.2.2 Algorithm setup

The algorithm setup in this experiment is as follows:

During each interaction, the heuristic query will be used after every 2 iterations. In case no users similar to the current one can be found, the default hybrid EGO Query will be used instead. This situation can happen when (1) no interaction has been recorded, or (2) no interaction has gone through that many iterations so that no previous user profile exists at that iteration, or (3) all similarities measured are negative, indicating a large discrepancy between the current user's preference and all previous users'. A random choice is made when multiple best candidates from $\mathscr{X}_s$ exist, i.e., more than one candidate has the most potential gain ($H1$) or the least potential cost ($H2$).

While the elicitation graph is updated once a new elicitation is done, the update of cost sets ($\mathbf{p}$) associated with leaf vertices will take place after every 1, 10 or 100 interactions during the experiment. Frequent update (e.g., updating after every elicitation) may take more advantage of the recorded elicitations, while delayed update (e.g., updating after every 100 elicitations) may form a better query strategy when more observations are collected at the cost of not being adaptive enough. We set up tests with different update frequencies in the experiment and observe their performance.

The initial query contains the pair $\{\mathbf{x}_1, \mathbf{x}_{100}\}$ for all interactions.

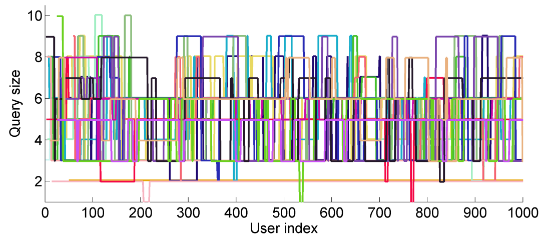
3.2.3 Experimental results

Figure 6 demonstrates the effects of $H1$ and $H2$ under different update frequencies. Simulated users enter the experiment in the same order across all different experiment configurations. The performance measure we report here is the number of queries used to reach optimally-preferred designs. For visualization purposes, the results are averaged over a span of 50 users and these average query sizes represent the effectiveness of the heuristic query algorithm at different stages of the experiment. The performance of the hybrid EGO Query algorithm without using knowledge from recorded elicitations is shown as a baseline. This performance is not constant throughout all the stages because each user group of 50 may contain a different combination of optimally-preferred designs.

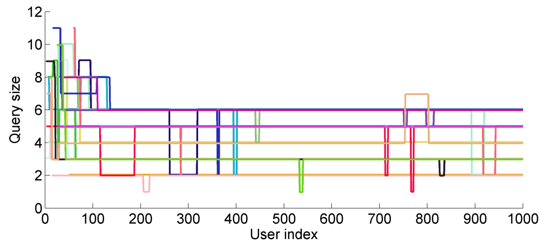
The baseline result is clearly outperformed by that of either $H1$ or $H2$. In addition, comparison between results from $H1$ and $H2$ favors $H2$ by a large margin. Not only does the average query size in $H2$ decrease more, it also achieves a more stable performance (around 4.5). A closer comparison of the two heuristics can be found in Figures 7(a) and 7(b) where the query sizes needed for each optimally-preferred design are tracked throughout the experiment. The elicitation graph is updated once per elicitation in these two figures. Clearly the query algorithm using $H2$ converges to an almost fixed strategy after interacting with around 200 users, while that using $H1$ fails to converge.

We now discuss the effect of update frequency on the more promising $H2$. Figures 7(b), 7(c) and 7(d) provide the query sizes needed for each optimally-preferred design during the experiment for the three different update frequencies. Examination of these figures shows that higher update frequency leads to

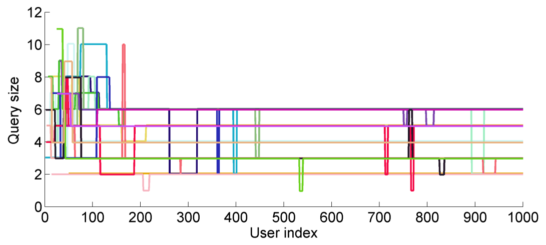
Copyright © 2012 by ASME



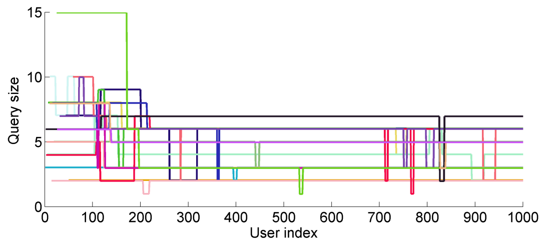
(a) $H1$, 1 user/update



(b) $H2$, 1 user/update



(c) $H2$, 10 user/update



(d) $H2$, 100 user/update

FIGURE 7. Performance comparison between $H1$ and $H2$. Each colored curve represents the query size required over the experiment for one optimally-preferred design defined as a grayed block in Figure 5. The elicitation graph is updated once per elicitation.

shorter adaptation period. In other words, the advantage of responding quickly to the accumulated observations overtakes its drawback of being misled by insufficient observations.

3.3 Initial query update ($H3$)

In addition to the above heuristics, it is intuitive that the initial query shall contain designs that are more "relevant" to the

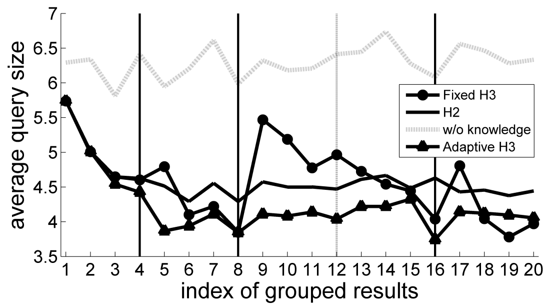


FIGURE 8. Demonstration of the effects of $H2$, fixed and adaptive $H3$ heuristics in the same simulation experiment as before.

users. To this end, we propose to update the initial query with the designs that are mostly preferred by previous users.

To verify this idea, we rerun the previous experiment, adding to $H2$ the functionality of updating the initial query after every 200 interactions. We call this heuristic "fixed $H3$". We discard the previously built elicitation graph along with cost sets on leaf vertices once the initial query is updated. This is plausible for two reasons: (1) The average cost on each leaf vertex derived from previous elicitations can be misleading when the initial query is changed, and (2) $H2$ may converge to a stable query strategy in a short run as observed in the previous discussion. Therefore we are not losing too much information by discarding previous records. Figure 8 compares the performance of fixed $H3$ and $H2$, both updating cost sets after every elicitation. The four vertical lines indicate the moments when the updates are programmed to happen and the three solid lines indicate when the initial queries are actually changed.

Examination of this figure reveals that the benefit of changing the initial guess is not instant. In fact, the performance always gets worse right after the changes, which is reasonable since the record has to be rebuilt. The benefit of updating the initial query eventually takes place as the average query size gradually goes below that of $H2$. With these observations, we propose an "adaptive $H3$" that keeps the current initial guess when no significant change occurs in the distribution of optimally-preferred designs. Formally, we group the leaf vertices into two clusters based on their variable values. For each cluster, we assume that the observed leaf vertices are realized from a normal distribution centered at $\bar{\mathbf{x}}_i$, which has a maximum likelihood estimation $\bar{\mathbf{x}}_i^*$ as the arithmetic center of cluster i. We then find two leaf vertices from each cluster that are closest to their corresponding cluster centers. Let the current initial guess be a set $\{\mathbf{x}_i\}_{i=1}^2$. We measure how suitable the current initial query (a pair of designs) is at representing the clusters of optimally-preferred designs:

Copyright © 2012 by ASME

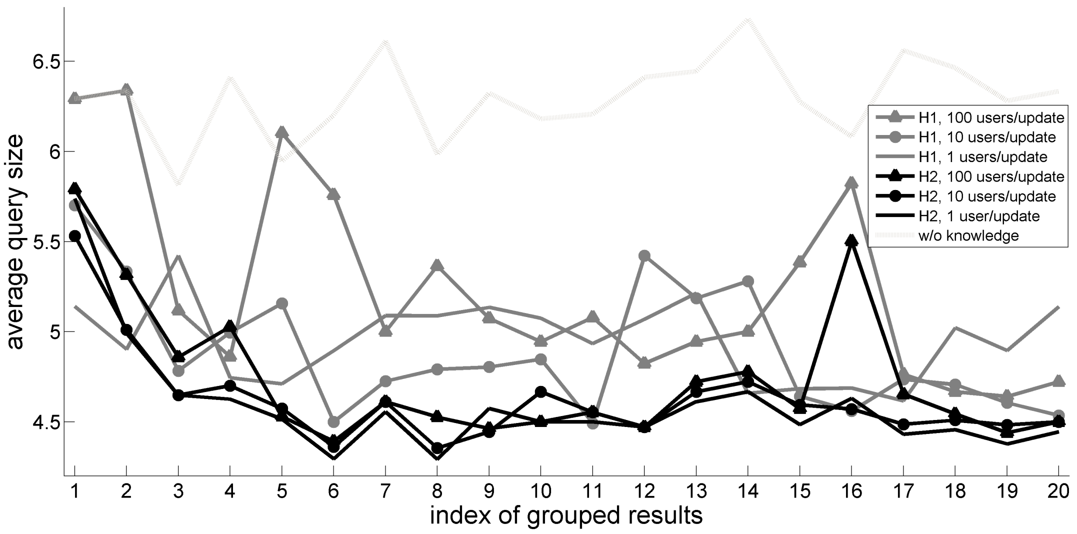


FIGURE 6. Demonstration of the effects of $H1$ and $H2$ in a simulation experiment. The horizontal axis represents 1000 simulated users coming into the interaction in a sequence. The vertical axis represents the average query sizes needed for a group of 50 users. Each curve illustrates the changing performance of the adaptive query algorithm with increasing knowledge about the crowd.

$$\triangle L = \left| \frac{L(\{\mathbf{x}_i^*\}_{i=1}^2) - L(\{\mathbf{x}_i\}_{i=1}^2)}{L(\{\mathbf{x}_i\}_{i=1}^2)} \right|, \tag{15}$$

where

$$L(\{\mathbf{x}_i^*\}_{i=1}^2) = \sum_{i=1}^{2} \sum_{j \in \text{cluster}_i} -||\mathbf{x}_j - \mathbf{x}_i^*||_2^2. \tag{16}$$

We set up this heuristic, the "adaptive $H3$", in a way that the above calculation is performed after every 100 elicitations, and the initial query is only updated when $\triangle L$ is over a threshold of 0.2. The performance of adaptive $H3$ is also shown in Figure 8. The improvement from the fixed update is visible and expected, since in this experiment there is no need to frequently change the initial query once we settle it close to the two cluster centers.

3.4 Discussion

In this subsection we compare the performance of $H2$ and adaptive $H3$ under a variety of experimental settings to verify their usage in different scenarios.

3.4.1 Changing trend Adaptive $H3$ can be useful in a situation where the distribution of optimally-preferred designs changes over time, simulating the changing trend in crowd preferences. To show this, an experiment was conducted where the users coming into the experiment will bear optimally-preferred designs that form moving clusters as shown in Figure 9. We compare the average query sizes needed in this scenario when using adaptive $H3$, $H2$ and the default algorithm in Figure 10. The improvement of adaptive $H3$ from other heuristics is significant.

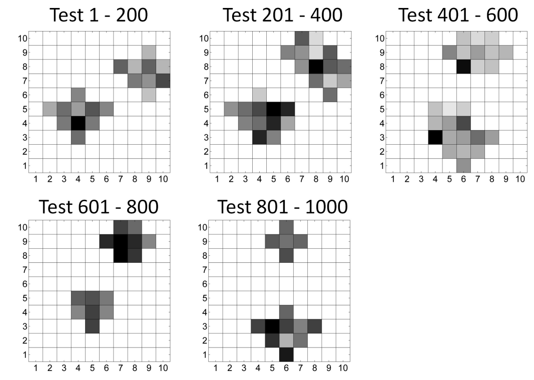


FIGURE 9. Densities of the optimal designs along time. The five figures show how the optimal designs are distributed at different stages of the experiment. The grey scale indicates the frequency of occurence.

3.4.2 Lack of clusters We examine the scenario where optimally-preferred designs from users are not clustered.

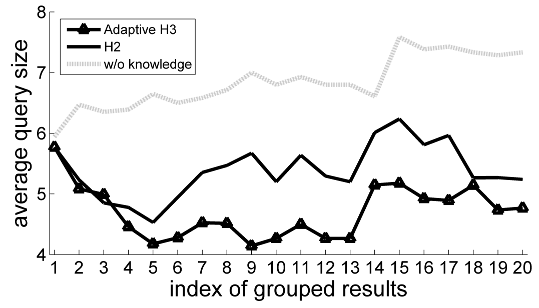


FIGURE 10. Demonstration of the effects of $H2$ and adaptive $H3$ heuristics in the changing trend experiment from Figure 9.

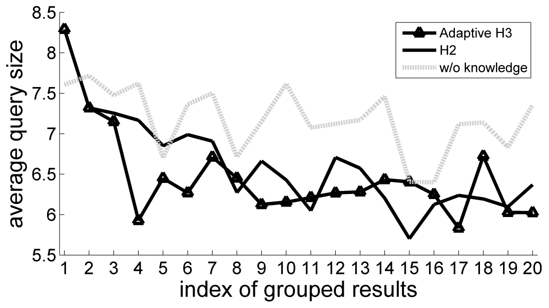


FIGURE 11. Demonstration of the effects of $H2$ and adaptive $H3$ heuristics when user preferences are diversified as shown in Figure 9.

Figure 11 shows the performance of $H2$, adaptive $H3$ and the default algorithm under a setup where optimally-preferred designs are uniformly distributed in $\mathscr{X}_c$. Here we only observe marginal improvement from the default algorithm by using either $H2$ and adaptive $H3$ as a heuristic. In addition, the difference between $H2$ and adaptive $H3$ is negligible, which leads to the implication that changing the initial guess is only valuable when the optimally-preferred designs are clustered.

3.4.3 Dimensionality of $\mathscr{X}_c$ Here we set up a candidate design set with 5 dimensions and 3 levels:

$$\mathscr{X}_c = \left\{ \mathbf{x} \in \mathbb{R}^5, x_{1,2,3,4,5} \in \{0, 0.5, 1.0\} \right\}. \tag{17}$$

Two experiments are conducted. One has optimally-preferred designs uniformly spread in $\mathscr{X}_c$ (no clusters) for the 1000 simulated users; the other has designs randomized from two clusters within $\mathscr{X}_c$ and each cluster contains three designs, i.e., the optimally-preferred designs of the simulated users are generated from the set of six pre-defined designs. Figures 12 and 13 compare the performance of adaptive $H3$ and $H2$ under the first and the second experiment settings, correspondingly. Figure 12 shows that

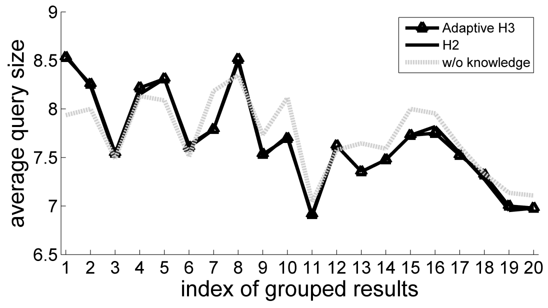


FIGURE 12. Demonstration of the effects of $H2$ and adaptive $H3$ heuristics when user preferences are diversified. Candidate set has 5 dimensions and 3 levels.

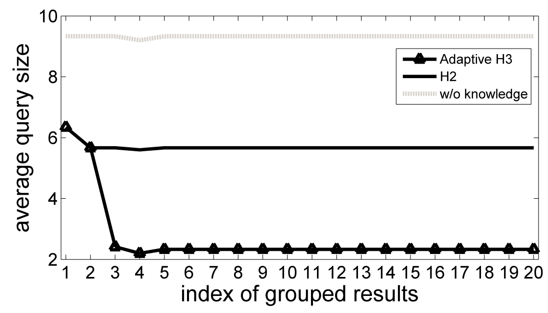


FIGURE 13. Demonstration of the effects of $H2$ and adaptive $H3$ heuristics when user preferences are clustered. Candidate set has 5 dimensions and 3 levels.

neither heuristic query algorithm will outperform the default algorithm when users' preferences are diversified in a large design space (high dimensionality of $\mathscr{X}_c$). On the other hand, Figure 13 shows that when users' preferences are clustered, the heuristic queries have better performance over time by taking advantage of the accumulated elicitation records, even when the dimensionality is high. As a matter of fact, the converged heuristic query algorithm from adaptive $H3$ in Figure 13 starts by querying the two cluster centers. Based on the user feedback, it picks another design belonging to the chosen cluster. Since there are only three possible guesses from each cluster, the maximum number of queries is three (including the last query with obvious answer). This adapted query strategy results in the average query size of 2.3 in the figure.

4 Related Work

The techniques presented in Section 2 are closely related to preference retrieval with implicit feedback [8]. The idea of enhancing interactions based on crowd information in Section 3 is inspired by recommender systems [1, 2]. In fact, the presented work can be considered an extension of a recommender system

Copyright © 2012 by ASME

in that we actively query users in order to optimize their preferences while a traditional recommender system passively provides recommendations on demand.

5 Conclusion and Future Work

Understanding preferences is important in many design creation activities. The drastic increase in consumer data and interaction opportunities encourage us to rethink the mechanism of capturing preference.

In this study, we defined "preference elicitation" as an interaction, consisting of a sequence of computer queries and human feedback, from where the user's most preferred design can be found. In order for the query algorithm to be efficient so that it fits in a short human-computer interaction, we introduced a solution that utilizes recent developments in information retrieval, recommender systems and global optimization. This proposed method has two components: First, the query algorithm is designed to retrieve and update a user preference model during the interaction, ensuring the next query to contain designs that are more likely to be preferred and different from existing ones. Second, the interaction is further shortened by incorporating information from previous elicitation records provided by users who share similar preferences to the current user.

Near-future work involves an examination on how erroneous human choice making and preference intransitivity will affect the proposed query algorithms. The role of the weight C in the retrieval problem P1 must be scrutinized to create preference models with good generalization performance. An enhancement in the heuristic query algorithm using crowd information is to incorporate user demographic data and explicit requests. Similarity can be more accurately measured when such data exist besides the retrieved user profile. A long-term extension of this work is to replace the vectorized design representation with a graph-based one. Similar to genetic programming where designs of different structures can be explored, introducing a graph representation of design enriches the possibility of design forms. Retrieval techniques that deal with structured data need to be investigated.

6 Acknowledgement

The authors are grateful for the advice and commentary provided for this work by Professor Richard Gonzalez, Department of Psychology, University of Michigan.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[3] O. Chapelle and Z. Harchaoui. A machine learning approach to conjoint analysis. *Advances in Neural Information Processing Systems*, 17:257–264, 2005.

[4] A. Chipperfield and P. Fleming. The MATLAB genetic algorithm toolbox. In *Colloquium Digest-IEEE*, pages 10–10. Citeseer, 1995.

[5] D. Cox and S. John. A statistical method for global optimization. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 1241–1246. IEEE, 1992.

[6] J. Hauser, O. Toubia, T. Evgeniou, R. Befurt, and D. Dzyabura. Disjunctions of conjunctions, cognitive simplicity, and consideration sets. *Journal of Marketing Research*, 47(3):485–496, 2010.

[7] C. Hoyle, W. Chen, B. Ankenman, and W. Nanxin. Optimal experimental design of human appraisals for modeling consumer preferences in engineering design. *ASME Journal of Mechanical Design*, 131(7):071008, 2009.

[8] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM, 2002.

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

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

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

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

[13] D. Kumar, C. Hoyle, W. Chen, N. Wang, G. Gomez-Levi, and F. Koppelman. 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*, pages 571–581, 2007.

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

[15] A. Lloyd. Threats to the estimation of benefit: are preference elicitation methods accurate? *Health Economics*, 12(5):393–402, 2003.

[16] E. MacDonald, R. Gonzalez, and P. Papalambros. Preference inconsistency in multidisciplinary design decision making. *ASME Journal of Mechanical Design*, 131(3):79–92, 2009.

Copyright © 2012 by ASME

[17] D. McFadden. Conditional logit analysis of qualitative choice behavior. *Frontiers in Econometrics*, pages 105–142, 1973.

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

[19] O. Netzer, O. Toubia, E. Bradlow, E. Dahan, T. Evgeniou, F. Feinberg, E. Feit, S. Hui, J. Johnson, and J. Liechty. Beyond conjoint analysis: Advances in preference measurement. *Marketing Letters*, 19(3):337–354, 2008.

[20] Y. Ren. *Design Preference Elicitation, Identification and Estimation*. PhD thesis, University of Michigan, Ann Arbor, Michigan, 2011.

[21] Y. Ren and P. Papalambros. A design preference elicitation query as an optimization process. *Journal of Mechanical Design*, 133:111004, 2011.

[22] M. Sanena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, Ann Arbor, Michigan, 2002.

[23] J. Secretan, N. Beato, D. D'Ambrosio, A. Rodriguez, A. Campbell, J. Folsom-Kovarik, and K. Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):373–403, 2011.

[24] K. Sims. Interactive evolution of dynamical systems. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 171–178, 1994.

[25] K. Sims. Galápagos. *Information online at: http://www.genarts. com/galapagos*, 1997.

[26] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

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

[28] H. Wassenaar, A. Sudjianto, J. Cheng, and W. Chen. Enhancing discrete choice demand modeling for decision-based design. *ASME Journal of Mechanical Design*, 127(4):514–523, 2005.

Copyright © 2012 by ASME