Accelerating Human-Computer Collaborative Search through Learning Comparative and Predictive User Models

Gregory S. Hornby
University of California Santa Cruz
NASA Ames Research Center, MS 269-3
Moffett Field, CA USA
gregory.s.hornby@nasa.gov

Josh C. Bongard
Department of Computer Science
University of Vermont
Burlington, VT USA
josh.bongard@uvm.edu

ABSTRACT

Interactive Evolutionary Algorithms (IEAs) are one of the few systems in which a human user and a computer algorithm are collaboratively working on a problem. To turn a basic IEA into the start of a Human-Computer Collaborative Computational system we have developed a system called The Approximate User (TAU). With TAU, as the user interacts with the IEA a model of the user's preferences is constructed and continually refined and it is this user-model which drives search. Here two variations of a user-modeling approach are compared to determine if this approach can accelerate IEA search. The two user-modeling approaches compared are: 1. learning a classifier which correctly determines which of two designs is better; and 2. learning a model which predicts a fitness score. Rather than having people do the user-testing, we propose the use of a simulated user as an easier means to test IEAs. Both variants of the TAU IEA are compared against a basic IEA and it is shown that TAU is up to 2.7 times faster and 15 times more reliable at producing near optimal results.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Design

Keywords

Evolutionary Design, Interactive Evolutionary Algorithm, Preference Learning, User Fatigue, User Modeling

1. INTRODUCTION

Interactive Evolutionary Algorithms (IEAs) are one of the few systems in which a human user and a computer algorithm are collaboratively working on a problem. An IEA is an interactive search algorithm that utilizes human input to make subjective decisions on potential problem solutions [9, 5, 13, 14]. In traditional interactive evolution, a human user is presented with one or more candidate individuals being evolved for selection. The human user directly performs selection and then the favored individuals are selected for propagation of offspring into the next generation. Current examples of this work on the Web are Picbreeder [12] and EndlessForms [7], both of which are based on using neural networks to encode 2D images (Picbreeder) or 3D shapes (EndlessForms).

Reliance on human input, however, induces a couple of major challenges. First, users suffer *user fatigue*: the quality and accuracy of human input greatly degrades with repeated prompts for input [14]. In addition, for typical non-interactive EAs, tens of thousands of evaluations are necessary to achieve interesting results, which is orders of magnitude more evaluations then can be expected from a single user. Finally, humans are generally far slower at evaluating designs than computer software is. To make IEAs viable, some method must be developed for overcoming these limitations of human users.

Given the limited number of human interactions possible with an IEA, the IEA must strive to make the most of what little data the user has provided. The approach we are analyzing is that of learning a model of the human user and using this model as the fitness function to drive the IEA. This idea came from the Estimation-Exploration Algorithm [2, 3], and was first applied to IEAs by Schmidt and Lipson [11]. The system we have developed is called **TAU**, for *The Approximate User*. Two different methods of user-modeling are described and compared: 1. learning a classifier which correctly determines which of two designs is better; and 2. learning a model which predicts a fitness score. Since the implementation used in this work is still in its preliminary stages, we hope that demonstrating a speedup on one domain is sufficient to show that it has potential. In addition, this work is a step toward turning an IEA into a truly Human-Computer Collaborative Search system where the strengths of each system are better utilized.

TAU is also a step toward a long term goal of making a human-computer collaborative search system. Traditionally, either a person does all the work or a computer search or optimization algorithm does all the work. With IEAs, a human is doing all the evaluations and the software is deciding what to be evaluated. We are working toward a system in which a human user is using their intelligence and experience to guide a search algorithm, taking advantage of the computer's speed to perform most of the evaluations on its own.

The rest of this paper is organized as follows. In Section 2 we review related work in IEAs and user-modeling, followed by a description of the TAU algorithm in Section 3. To demonstrate that TAU algorithm can accelerate IEA search we use a simulated human user, which is described in Section 4. The setup for the experiments is described in Section 5. Then in Section 6 we present experimental results which show that fitness-predictive user modeling accelerates IEAs over that of a basic IEA, although are not as fast as an IEA accelerated with comparative user modeling. Finally, we present our conclusions in Section 7.

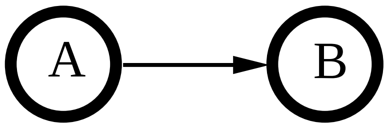
2. BACKGROUND

Quite recently there has been promising initial work toward addressing the user fatigue problem. One approach that has been used is to hardcode mathematical heuristics of aesthetics, and this has found some success for the interactive evolution of jewelry [15]. This system has several heuristics of beauty built in and reduces the amount of feedback needed from the user by two orders of magnitude. It can be thought of as a hybrid approach in which evaluations are partially done by an encoded fitness function – the heuristics of aesthetics – and partially done by the human user. Limitations of this approach are that it still requires a hard-coded fitness function and that results are somewhat dependent on it. Of interest are approaches in which there is no such dependence on a hard-coded fitness function.

An alternative to hard-coding heuristics is to build a model of the user's preferences with ideas from statistical Machine Learning. One system is to treat user feedback as inputs to a traditional parameter estimation system [1]. This leverages the speed of existing statistical machine learning systems but is limited to parameterized design spaces. To move beyond parameterized encodings, another approach is to learn weights on grammatical rules for constructing a design [6]. While allowing for search through a topological space of designs, this does not scale to systems with large sets of rules, or which require large derivation trees to produce a design, or in which multiple sets of rules can produce acceptable solutions.

Our approach is to build a model of what the user wants to drive search, and to continuously learn and refine this model of the user's preferences concurrent with the design process. The idea behind our approach comes from prior work with the Estimation-Exploration Algorithm [2, 3], in which a coevolutionary system is used to evolve an *estimation* population, which evolves improvements to models of the hidden system, given pairs of input/output data obtained from the physical model(s) being approximated; and an *exploration* population, which evolves intelligent tests to perform on the hidden target system using the best models so far. In this case, the "hidden system" is the human user, of whom the computer is trying to build a model.

By having a computer model of the human's desires, this model can be used tirelessly to perform thousands or millions of evaluations and thereby circumvent the limitations of having human users act as the fitness function. Already a version of this approach has been tried on IEAs and it seemed to work well [11]. We have implemented our own variant, which we have called The Approximate User (TAU), with two ways of modeling the user and here we are performing a more rigorous comparison of this approach against a basic IEA.



$$\begin{tikzcd} A \arrow[r] & B \end{tikzcd}$$

Figure 1: A simple relations graph showing that design `A` is preferred over design `B`.

3. OVERVIEW OF TAU

The TAU algorithm differs from a Basic IEA in that it uses a model of the user to perform its fitness evaluations rather than having the user manually evaluate each candidate solution. A user-model is built from a relations graph, which is a directed graph which stores every preference provided as input by the user. From this relations graph, modern machine learning techniques are used to train a model which can accurately match the user's preferences stored in this graph. This model of the user is then used as the evaluation function for a traditional optimizer to create a new set of solutions. Once a new set has been produced, a subset of them are presented to the user and the process repeats until a satisfactory result is produced.

An initial version of The Approximate User (TAU) algorithm for user-modeling has been implemented and an IEA augmented with TAU operates as follows:

1. Use the existing User Model to generate a set of candidate designs to present to the user. If the User Model is empty, generate random designs. This set of designs should be both good and diverse.

2. After the user has indicated their preference, update the relations graph by inserting the designs which were presented along with the user's preferences.

3. Do one of 3a (Comparator) or 3b (Fitness-Predictor):

    3a. Create a newer User Model by training a classifier to correctly predict each relation in the relations graph.

    3b. Sort the designs into layers of preferences, assign a fitness to each layer and then create a newer User Model by training a function approximator to match this.

4. Quit if a satisfactory solution has been produced.

For Step 1, we have implemented an Evolutionary Algorithm (EA) to create a set of designs using the User Model from Step 3 to either rank (3a) or score (3b) individuals. The EA used is configured to have a population size of 5 times the size of the grid shown to the user and a generational EA is run for 25 generations. Individuals are selected using tournament selection, with a tournament size of 2. New individuals are created using either mutation (65% of the time) or recombination (35% of the time) and are inserted into the population using Deterministic Crowding [10] to maintain genotypic diversity.

The data structure from which user models are built is the *relations graph.* The relations graph (Step 2) is a directed graph in which each node represents a design which has been shown to the user and each edge represents a user preference. For example, if a user is shown designs `A` and `B` and indicates that s/he prefers design `A`, then the relations

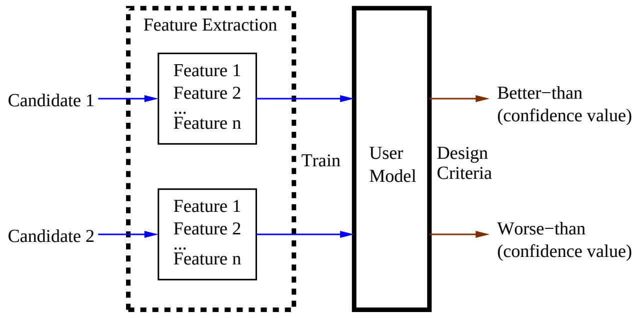


Figure 2: The basic structure of a Comparator user model. Features are extracted from two candidate designs and are fed into the Comparator, which then uses these features to predict which one is better.

graph will have two nodes, `A` and `B`, and a directed edge from `A` to `B` (Figure 1). The difference between the two variants of TAU is in how this relations graph is used to model a user.

The first variant of TAU models a user with a *Comparator* (Step 3a.), which takes the features of two designs as inputs and as an output it indicates which design is preferred (Figure 2). This user model is trained on the relations in the relations graph to predict, for any pair of designs, which one is better. In determining dominance from the relations graph, it is assumed that the dominance relation is transitive. Consider a population of candidate solutions (called *individuals* by those in the field of Evolutionary Computation) ind_A, ind_B and ind_C. If ind_A is better than ind_B, and ind_B is better than ind_C, it is assumed that ind_A is also better than ind_C.

Thus two key advantages of the Comparator approach are that more relations can be derived from the graph than there are elements in the graph, and the number of relations grows faster than the size of the graph. For example, the relations graph in Figure 3(a) contains six individuals (`A` through `F`) and represents a subset of the entire relations graph that might exist after a couple of prompts to the user. There are nine relations that can be derived from this graph – indicating the user's preference from past queries – with the first five relations being the arrows that are shown. The rest are: `A` is better than `E`; `A` is better than `F`; `C` is better than `E`; and `C` is better than `F`. If, after being presented with new designs `G` and `H` and previously seen designs `B` and `C`, the user indicates that s/he prefers `B` and `G` (Figure 3(b)), then two new nodes and four edges are added to the relations graph (Figure 3(c)). In addition to this, four additional relations can be derived: `G` is preferred over `D`, `E` and `F`, and `A` is preferred over `H`. It is this set of relations which can be derived from the relations graph which the Comparator user-model is trained to match.

The comparator user-model used here is quite similar to that of Schmidt and Lipson [11] with a few key differences. In Schmidt and Lipson's implementation, a population of ANNs was evolved using EAs, and the Estimation-Exploration Algorithm was used to determine a pair of designs to present to the user to evaluate and indicate their preference. Here, an ensemble of ANNs is trained using backpropagation and several designs are presented to the user with designs selected by using the ensemble to identify the better candidates in the population.

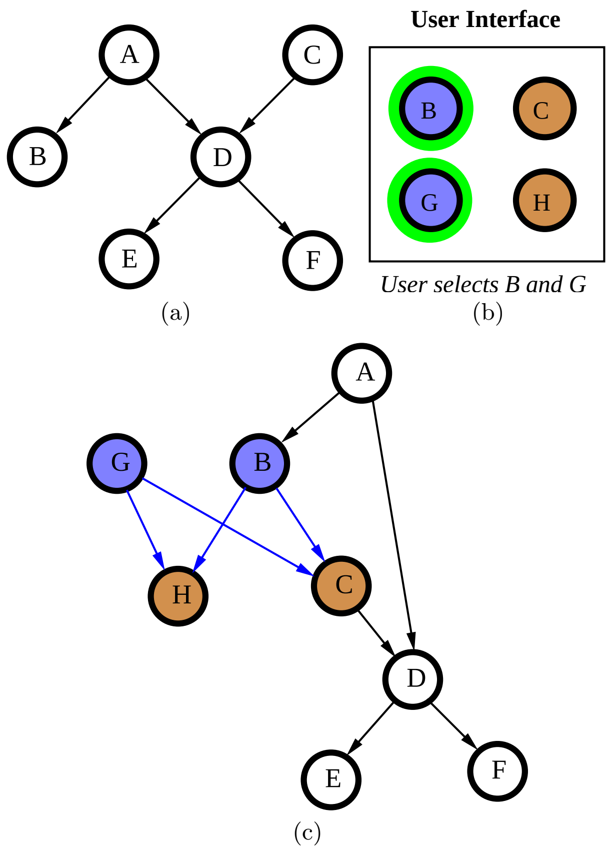


Figure 3: An example of how an update of the relations graph works: (a) shows the current relations graph consists of candidate solutions A through F; (b) two of these "old" candidates are shown to the user as well as two new ones, G and H, of which the user selects B and G; (c) based on the user's selection, the relations graph is now updated.

The other approach to user modeling is that of fitness prediction (Step 3b.). With this approach, the nodes in the relations graph are organized into layers of dominance, a fitness value is assigned to the nodes in each layer, and function approximator is trained to match the features of the designs to the appropriate fitness value. To put nodes into layers, all nodes which are not dominated by any other nodes are put into a layer and removed from the graph. This step of removing all non-dominated nodes to create a layer is repeated until all the nodes are removed from the relations graph. Applying this method to the relations graph in Figure 3(c) produces the layers in Figure 4.

To implement both user models the Fast Artificial Neural Network (FANN) library is used. ANNs are used because they have robust regression power with excellent interpolation and extrapolation characteristics [8]. Their classification output also corresponds to their statistical confidence in their prediction. In other words, noisy samples or conflicting samples reduce prediction confidence but in general maintain prediction accuracy [4]. The basic structure of a comparator neural net is shown in Figure 2. To improve performance, we are using an ensemble of three ANNs to

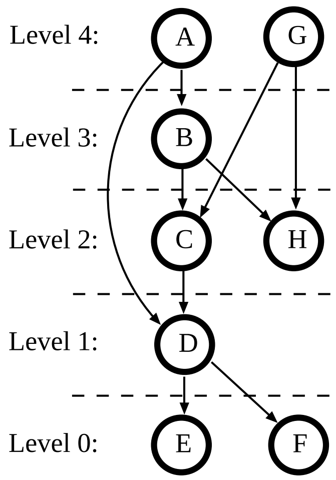


Figure 4: The result after taking the relations graph in Figure 3(c) and assigning levels to each node.

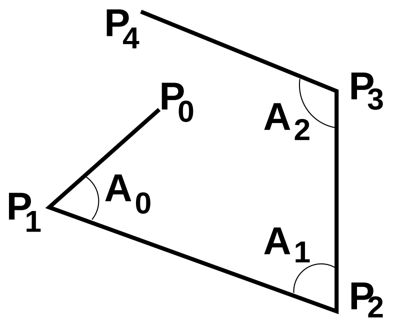


Figure 5: An example drawing that is scored for its closeness to a square.

create a User Model. Each ANN in the ensemble has a single hidden layer of 7 hidden units, a 40% connection density of the available, with weights randomly selected in the range of -0.1 to 0.1. Each ANN is trained using backpropagation for at most 50 iterations through the training data or until the training error is less than 0.001.

4. SIMULATING A HUMAN USER

Instead of performing comparison studies with real people, in this work a simulated person is used to drive the IEAs. The advantage of a simulated person is that comparisons between different versions of the TAU IEA and the Basic IEA can be done as often as desired. The simulated person is implemented as a combination of a scoring function for how well candidate designs match a target shape and a method for using this to drive the IEA. This scoring function could be used as a fitness function in a regular EA but in this case it is used as the target function which the TAU algorithm is trying to learn. Since it is not directly used to drive the EA we choose not to call it a fitness function to avoid confusion.

Here, designs are constructed from a contiguous sequence of connected line segments. The design-scoring function used by the simulated human user takes as input the endpoints of these line-segments – P_0, P_1, P_2, … – and computes how close these line segments are to the target shape. For these experiments, the test problem used is that of creating a square out of a sequence of four connected line segments. The four line segments are contiguous and are encoded as a real-valued vector with five x and y coordinates for the 5 end points, P_0, P_1, … P_4, Figure 5. This scoring function has three distinct components: a score on the gap between P_0 and P_4; a score on how similar in length each line segment is to each other; and a score on how similar each of the three angles, A_0, A_1 and A_2 is to a right angle. Each of these three scoring-components has a range of 0 to 1 and the overall score is a product of these three sub-functions.

When creating a square from the five points which specify four contiguous line segments the first and last points, P_0 and P_4, must be the same. To score for this result the gap score function, S_{gap}, divides the distance between points P_0 and P_4 by the sum of the lengths for each of the four line segments:

$$S_{gap} = 1 - \frac{\text{distance from } P_0 P_4}{\text{sum of the length of all line segments}} \quad (1)$$

Another characteristic of a square is that all four sides have the same length. The score for this, the sub-function $S_{lengths}$ takes the length of each line segment and divides this by the average length of the four line segments. The ratio of the length of each side to the average length, L_i, is compared against the desired ratio, $L_{desired}$. For a square $L_{desired}$ is 1 for all line segments. To create a value between 0 and 1, the smaller of these two values is divided by the larger. This is done for each line segment and all four of these values are multiplied together:

$$S_{lengths}(L_{desired}) = \prod_{i=0}^{4} \begin{cases} \frac{|L_{desired}|}{|L_i|} & \text{if } L_i > L_{desired} \\ \frac{|L_i|}{|L_{desired}|} & \text{otherwise} \end{cases} \quad (2)$$

The third characteristic that is scored for is to score for the angles being right angles using the sub-function S_{angles}. Here, each angle A_i is compared against the desired angle, $A_{desired}$ and a value between 0 and 1 is computed similar to with $S_{lengths}$. For a square $A_{desired}$ is either $-\pi/2$ for all three angles or $+\pi/2$ for all three angles. The result for all three angles is multiplied together and returned:

$$S_{angles}(A_{desired}) = \prod_{i=0}^{2} \frac{|A_{desired}|}{|A_i - A_{desired}| + |A_{desired}|} \quad (3)$$

The overall score for how well a given shape matches the target shape is a product of the previous three sub-functions. For a square, the two options are for all line segments to have the same length and all three angles must be positive 90° turns ($\pi/2$) or they must all be negative 90° turns ($-\pi/2$):

$$S = S_{gap} S_{lengths}(1) max\big(S_{angles}(-\pi/2), S_{angles}(\pi/2)\big) \quad (4)$$

This approach to scoring how well a given shape matches a target shape is generic and can be used for scoring for different shapes by supplying the desired line-segment ratios and angles for the target shape. Because of symmetries, there are often multiple ways of producing a given shape from a sequence of line segments so the function returns the maximum of the different options.

In addition to the scoring function, the simulated user has a simple algorithm for driving the IEA to try and produce a desired drawing. The user interface for the IEA allows the user to: select and de-select designs; create a new set of candidate designs based on the current selections; discard all of the existing designs and create a new set of candidate designs based on the previous selections; and back-track to the previous set of designs. Roughly, the algorithm for the simulated human user requests a set of randomly generated designs until it finds one that scores higher than 0.25. It then iterates over selecting the top design, along with up to 2 other drawings which have a score within 10% of the top drawing. If the top drawing is not as good as the best drawing from the previous iteration, the algorithm requests an alternative set of candidates. If after three tries a new best drawing is not found, it backs up a level and tries again.

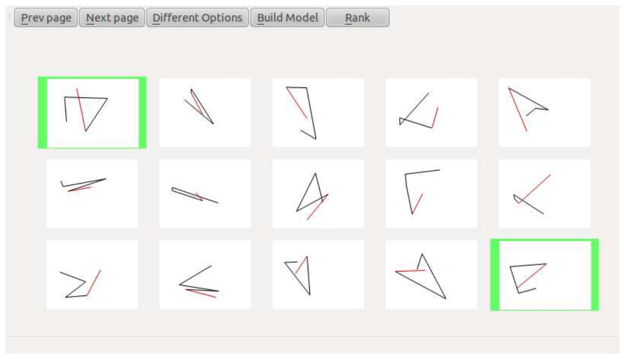
More precisely, the algorithm for the simulated user is as follows:

```
1: Level = 0
2: Set existing best to 0.
3: repeat        ▷ Starting with randomly generated designs.
4:     Request new random designs.
5:     Score each design.
6: until Best score is > 0.25
7: repeat
8:     if New best > previous best. then
9:         Level = Level + 1
10:        Tries[Level] = 0
11:        Select the best design.
12:        Select next 2 best with score within 10% of best.
13:        Submit selections and request new designs.
14:        Score each design.
15:        continue.
16:    else
17:        Tries[Level] = Tries[Level] + 1.
18:        if Tries[Level] > 3 then
19:            Go back to previous generation.
20:            Level = Level - 1
21:            if Level = 0 then
22:                Goto line 1.        ▷ Move to initial state.
23:            end if
24:        end if
25:        Request an alternative set of design.
26:        Score each design.
27:    end if
28: until Best score is 1.
```

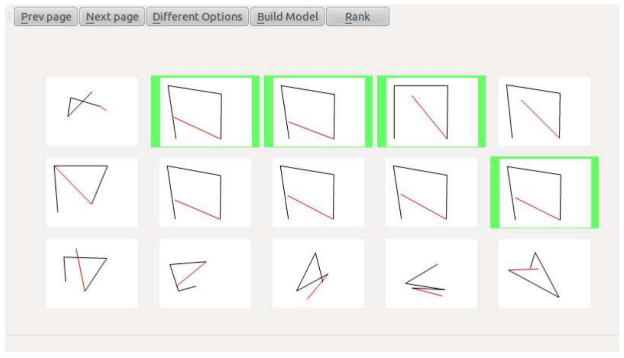
5. EXPERIMENTAL SETUP

To demonstrate that concurrent construction of a user-model can accelerate search we compare the TAU algorithm – using both the Comparator (TAU-C) and Fitness Predictive (TAU-FP) approaches to user modeling – against a Basic IEA. The Basic IEA has the designs selected by the user being the parent designs for creating the next set of designs to present to the user. The configuration of TAU was described in Section 3, and both types of user modeling were tested. All of these IEAs are driven by the simulated human user described in Section 4.

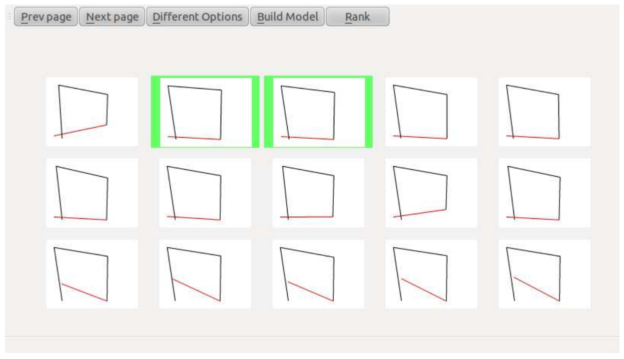
An example of the application with a 3x5 grid of designs is shown in Figure 6. This has a sequence of pages shown to the user in interactively designing a square. The top page (Figure 6(a)) shows an initial set of randomly generated line drawings along with the simulated user's selections (in green). The next three images (Figures 6(b) to 6(d)) show the subsequent iterations of presenting the [simulated] user with new designs and their selections. In this exam-



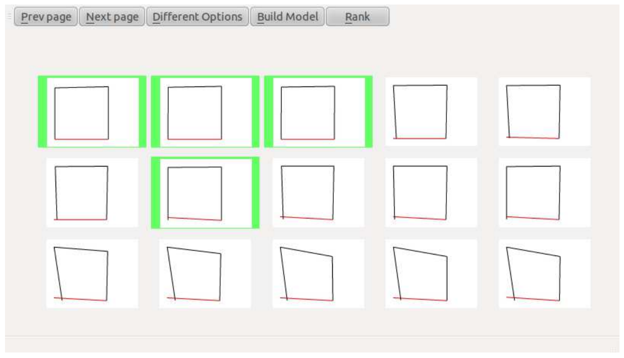
(a)



(b)



(c)



(d)

Figure 6: A sequence of pages in interactively designing a square using the TAU Fitness Predictive user-modeling system.

ple the TAU algorithm is used with the fitness predictive user-modeling system.

An important point to note is that with TAU, the features being used to build and utilize user-models are different from the features used by the simulated human user to score designs. For the TAU IEAs, the features for a given design have the ratio of the length of each line segment to the average length of line segments in that design and the angle between each line segment. Also, the angle of the first line segment to the horizon is included to indicate the overall orientation of the design and the size of the average line segment is included to provide an indicator for the overall size of the drawing.

6. EXPERIMENTAL RESULTS

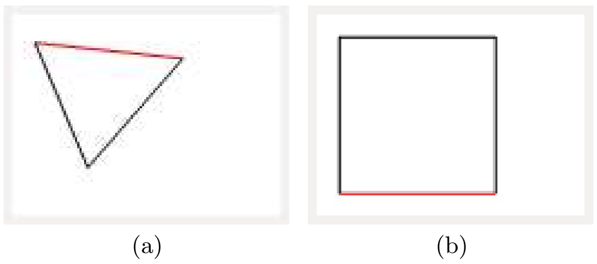


Figure 7: The two test problems tried are to interactively evolve an equilateral triangle (a) and a square (b).

For these experiments 250 trials were run with each of the Basic IEA, and TAU with both the Comparator (TAU-C) and the Fitness Predictive (TAU-FP) user models on grid sizes of 3x3, 3x4 and 3x5. These were tried on two test problems: 1. producing an equilateral triangle; and 2. producing a square (Figure 7). Results of our experiments for the 3x3 and 3x5 grid sizes are shown graphically in Figure 8 and also in tabular form for the 3x5 grid in Table 1. Results for the 3x4 grid are not shown, but for all three systems these fall roughly in between results for the 3x3 and for the 3x5.

The graphs in Figure 8 show that all variants of IEAs perform better with a larger grid size and do better on the easier problem (creating a triangle out of three lines) than on the harder problem (creating a square out of four lines). The left set of graphs shows the frequency (out of the 250 trials) in which a given IEA reached a given degree of optimality in the design problem. In all cases TAU-C had the greatest likelihood of success, and only had a steep dropoff after the 95% range. The TAU-FP approach was not as reliable and tended to have a moderate dropoff in likelihood to succeed starting at around the 35% range, with a steep dropoff happening around 90% to 95%. Both variants of TAU were able to achieve 100% designs (75% of the time with TAU-C and 33% of the time with TAU-FP). The least reliable is the Basic IEA which, while more reliable than TAU-FP up to around 85%, then had a large drop in reliability and struggled to make even 98% of optimal with any regularity.

The set of graphs on the right side of Figure 8 show another important difference between TAU and the Basic IEA. With the Basic IEA, the number of selection iterations needed to achieve a given level of optimality seems to be growing at a polynomial rate. Combine this with the Basic IEA's rapidly decreasing success rate and it suggests that the Basic IEA will not be able to scale to achieving good results on more challenging problems. In contrast, with the TAU IEA it seems that a few selection rounds are needed for the algorithm to learn a reasonable model of what the user wants and, after this, it has a fairly flat, linear growth in the number of selection rounds needed to achieve a given level of design optimality. Here, TAU-C is about twice as fast as TAU-FP, and both have a slight upward inflection in the number of search iterations to get the last few percent of optimality. Overall, both variants of TAU show accelerated search over the basic IEA.

7. CONCLUSION

In this paper the method of accelerating IEAs through user-modeling was analyzed with The Approximate User (TAU) algorithm. With TAU, a relations graph of the user's preferences is updated after each interaction and this relations graph is used to build either a Comparative or Fitness Predictive model of the user. It is these user models which are used to drive an EA, thereby accelerating search by allowing many search evaluations to take place much more rapidly than by a human user.

To validate the effectiveness of TAU, we developed an artificial human user to drive search using a basic IEA and with the TAU IEA. Experiments were run comparing a Basic IEA against both variants of TAU. Overall, TAU was up to 15 times more reliable at achieving near optimal results and and more than 75 times more reliable at achieving optimal results. In addition to being more reliable, TAU achieved near-optimal results up to 2.7 times faster. In the best case, some trials with the TAU IEA achieved an optimal designs in 4-8 selection iterations. With a much a higher success rate at finding good solutions and the ability to find these solutions faster, the TAU algorithm has considerable promise in overcoming user fatigue.

We expect that by developing better approaches to modeling a human user's preferences that the TAU IEA can be made faster and able to scale to more difficult problems. In addition, this work is a step toward turning an IEA into a truly Human-Computer Collaborative Search system where the strengths of both the human and the computer are better utilized.

Acknowledgments.

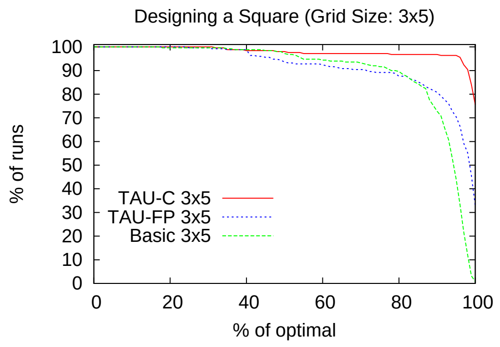
This research was supported in part by the NSF Creative-IT grant 0757532 and DARPA M3 grant W911NF-1-11-0076. In addition, thanks to Grace Lin for her work on an early prototype.

8. REFERENCES

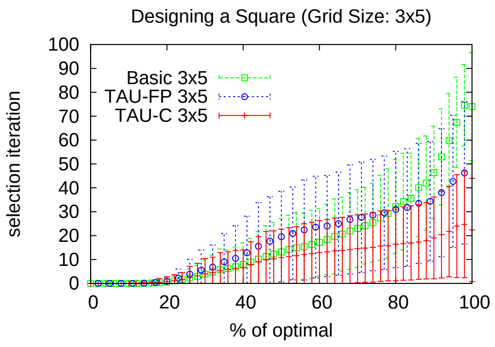
[1] G. J. Barnum and C. A. Mattson. A computationally assisted methodology for preference-guided conceptual design. *Journal of Mechanical Design*, 132, 2010.

[2] J. Bongard and H. Lipson. Nonlinear system identification using coevolution of models and tests. *IEEE Transactions on Evolutionary Computation*, 9:361–384, 2005.

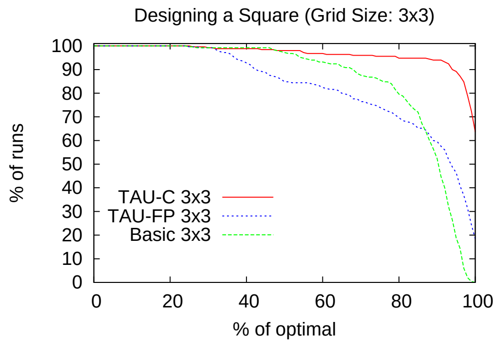
[3] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104:9943–9948, 2007.



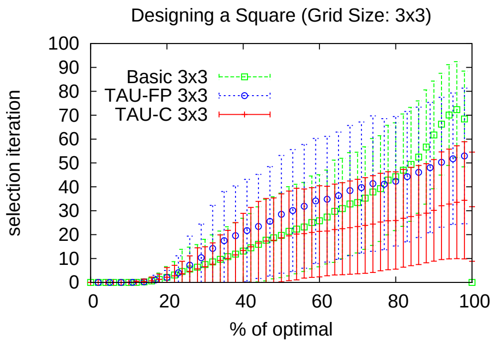
(a)



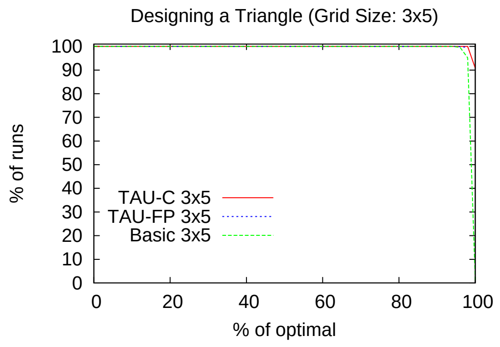
(b)



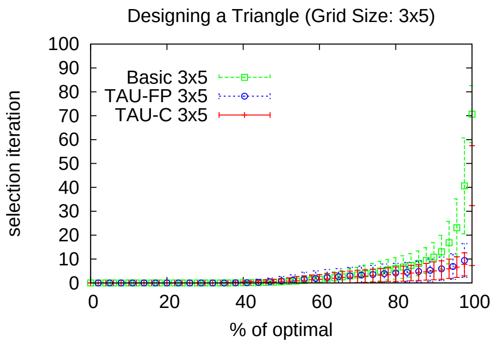
(c)



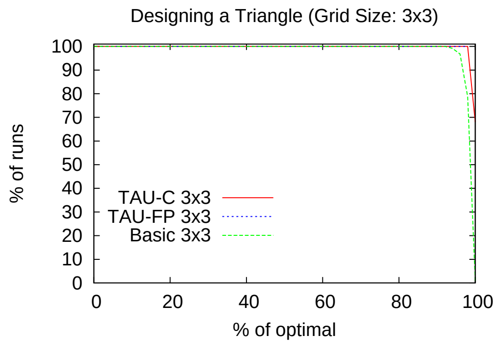
(d)



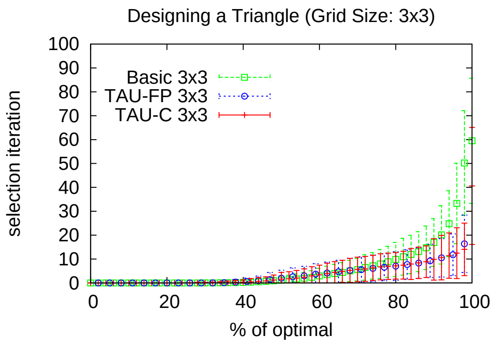
(e)



(f)



(g)



(h)

Figure 8: This figure shows graph which have plots of the performance of the Basic IEA and TAU using both a Comparator user-model (TAU-C) and a Fitness-Predictive user-model (TAU-FP). The graphs on the left show the probability of reaching a given percentage of optimal and the graphs on the right are the expected number of iterations to reach a given degree of optimality. This shows that the Comparator user models are better than the Fitness-Predictive ones, and both are better than the Basic IEA.

	Square					
	Basic IEA		TAU Fitness-Predictive		TAU Comparator	
% of Optimal	Prob Success	Avg Iterations	Prob Success	Avg Iterations	Prob Success	Avg Iterations
80	0.896	32.317 ± 19.8847	0.876	30.9726 ± 24.3431	0.968	16.4339 ± 15.2793
90	0.728	46.544 ± 19.4014	0.808	35.604 ± 25.29	0.968	19.2727 ± 17.2324
95	0.44	62.7455 ± 20.1492	0.704	42.8182 ± 27.6687	0.964	22.971 ± 19.9527
98	0.112	73.5357 ± 16.8907	0.54	45.7111 ± 29.7705	0.904	24.7522 ± 22.3901
99	0.028	68.5714 ± 12.6867	0.456	46.2632 ± 30.6672	0.836	24.8612 ± 23.7368
100	0.008	74 ± 22.6274	0.328	41.5732 ± 30.1236	0.756	22.3704 ± 21.6112

	Triangle					
	Basic IEA		TAU Fitness-Predictive		TAU Comparator	
% of Optimal	Prob Success	Avg Iterations	Prob Success	Avg Iterations	Prob Success	Avg Iterations
80	1	6.084 ± 4.54042	1	4.2 ± 4.04294	1	3.672 ± 3.07151
90	1	10.836 ± 6.15864	1	5.46 ± 4.76352	1	5.084 ± 3.74554
95	1	19.452 ± 10.1443	1	6.896 ± 5.3283	1	6.168 ± 4.0105
98	0.952	41.0252 ± 20.315	0.996	9.42972 ± 7.0628	1	7.828 ± 4.81531
99	0.696	57.431 ± 22.2958	0.996	13.2008 ± 9.83287	1	9.4 ± 5.88579
100	0.012	70.6667 ± 11.9304	0.732	43.8525 ± 25.7719	0.908	32.37 ± 25.087

Table 1: Summary of results for evolving a square on the 3x5 grid using the Basic IEA and the TAU algorithm with both Fitness-Predictive user-modeling and Comparative user-modeling. The top shows the results on interactively evolving a square, and the bottom are results on evolving a triangle. Both variants of TAU are more than 15 times more likely to reach 99% of optimal than the Basic IEA and up to 2.7 times faster.

[4] J. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fogelman-Soulie and Herault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, NATA ASI Series. Springer, 1990.

[5] C. Caldwell and V. S. Johnston. Tracking a criminal suspect through 'face-space' with a genetic algorithm. In R. K. B. L. B. Booker, editor, *Proc. of the Fourth Intl. Conf. on Genetic Algorithms*, pages 416–421, San Mateo, CA, 1991. Morgan Kaufmann.

[6] M. I. Campbell, R. Rai, and T. Kurtoglu. A stochastic graph grammar algorithm for interactive search. In *14th Design for Manufacturing and the Life Cycle Conference*, pages 829–840. ASME, 2009.

[7] J. Clune and H. Lipson. Evolving three-dimensional objects with a generative encoding inspired by developmental biology. *Lecture Notes in Computer Science*, 2011.

[8] G. Cybenko. Approximations by superpositions of a sigmoidal function. *Math. Contrl., Signals, Syst.*, 2:303–314, 1989.

[9] R. Dawkins. *The Blind Watchmaker*. Harlow Longman, 1986.

[10] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 27–36. North-Holland, 1992.

[11] M. Schmidt and H. Lipson. Actively probing and modeling users in interactive co-evolution. In M. K. et al., editor, *Proc. of the Genetic and Evolutionary Computation Conference, GECCO-2006*, pages 385–386, Seattle, WA, 2006. ACM Press.

[12] J. Secretan, N. Beato, D. B. D. Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 2011.

[13] K. Sims. Artificial Evolution for Computer Graphics. In *SIGGRAPH 91 Conference Proceedings*, Annual Conference Series, pages 319–328, 1991.

[14] H. Takagi. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. In *Proceedings of the IEEE*, pages 1275–1296, 2001.

[15] S. Wannarumon, E. L. J. Bohez, and K. Annanon. Aesthetic evolutionary algorithm for fractal-based user-centered jewelry design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22:19–39, 2008.